Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Aleš Kurent

Sledenje spletnim uporabnikom z zajemom prstnega odtisa naprave

diplomsko delo

univerzitetni študijski program računalništva in informatike

Mentor: izr. prof. dr. Zoran Bosnić

Ljubljana, 2014

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika dela:

Sledenje spletnim uporabnikom je danes razširjena praksa, ki se najpogosteje izkorišča za profiliranje uporabnikovih interesov in v komercialne namene (oglaševanje). Tehnike sledenja se razvijajo že 20 let in postajajo tudi vse bolj zanesljive ter s tem odporne na posege uporabnikov.

V diplomski nalogi naj kandidat naredi pregled pristopov za sledenje, pri čemer naj se osredotoči na postopek sledenja z zajemom prstnega odtisa naprave. V diplomskem delu naj razvije svoj sistem in ga evalvira na izbrani množici uporabnikov in naprav.

Izjava o avtorstvu diplomskega dela

Podpisani Aleš Kurent, z vpisno številko **63070126**, sem avtor diplomskega dela z naslovom:

Sledenje spletnim uporabnikom z zajemom prstnega odtisa naprave

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom izr. prof. dr. Zorana Bosnića,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 28. avgusta 2014 Podpis avtorja:

Zahvala

Zahvaljujem se vsem prijateljem, ki so mi tekom študija stali ob strani, mi pomagali in se skupaj z mano smejali.

Rad bi se zahvalil tudi mentorju, izr. prof. dr. Zoranu Bosniću, za pomoč pri izdelavi diplomske naloge ter vsem, ki so sodelovali v moji raziskavi.

Posebej pa bi se rad zahvalil moji družini za potrpežljivost in podporo ter moji punci Kaji, ki me je vedno spodbujala in bodrila.

Hvala.

Mojim najbližjim.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
HTTP	Hypertext Transfer Protocol	protokol za izmenjavo hiperteksta
IP	Internet Protocol	internetni protokol
NAT	Network Address Translation	prevajanje spletnega naslova
ISP	Internet Service Provider	ponudnik spletnih storitev
XSS	Cross-site scripting	večdomensko izvajanje kode
CPE	Central Processing Unit	centralna procesna enota
RTC	Real-Time Clock	ura realnega časa
OS	Operating System	operacijski sistem
JSON	JavaScript Object Notation	notacija objekta JavaScript

Povzetek

Sledenje spletnim uporabnikom je neizogiben del spleta. Različne metode sledenja so na spletu prisotne že od samega začetka. Najbolj uporabljena tehnologija, piškotek HTTP, je stara že 20 let, kot kaže pa bo na tem mestu ostala še vrsto let. Z razvojem spleta in spletnega oglaševanja so se spremenile tudi metode sledenja uporabnikom. Postajajo vse bolj agresivne, vsiljive in neopazne, zato so se problematike lotili tudi zakonodajalci.

V našem diplomskem delu smo proučili eno od naprednih metod sledenja z zajemom prstnega odtisa naprave. Razvili smo sistem zajema prstnega odtisa in prepoznave uporabnika, ki ne uporablja piškotkov HTTP. Uspešnost našega sistema smo preverili v raziskavi na realni množici uporabnikov, kjer smo dosegli 98,7% uspešnost prepoznave.

Ključne besede: spletno sledenje, prstni odtis naprave, prstni odtis brskalnika.

Abstract

Tracking of web users is inevitable part of the World Wide Web. Different methods of web tracking exist from the very beginning of the World Wide Web. Well-known and broadly used method of web tracking, called the HTTP cookie, is a technology that is more than 20 years old and it seems it is here to stay for many years to come. With the growth of Internet and online advertising, methods of web tracking are also changing and improving. As they are becoming more aggressive, intrusive and invisible, lawgivers started to tackle the rising problem.

In our thesis, we have examined one of the advanced methods of web tracking, called device fingerprinting. We have developed a device fingerprinting system that can track and recognize web users without the use of HTTP cookies. Performance of our system was tested in a study on a real set of web users, where our system correctly recognized 98.7% of users.

Keywords: web tracking, device fingerprint, browser fingerprint.

Poglavje 1

Uvod

Z razmahom svetovnega spleta smo postali informacijska družba. Velikanska množica podatkov je dosegljiva sleherniku, ki ima dostop do svetovnega spleta, medčloveška komunikacija pa je postala hitrejša in enostavnejša. Za mnoge je splet postal nepogrešljiv del vsakdana. Običajna opravila, kot so pošiljanje pošte, nakupovanje, izobraževanje in še mnogo drugih, se vse bolj premikajo v digitalni svet. Splet je zaradi tega postal zanimiv tudi za oglaševalce. Spletne storitve in oglaševalci želijo poznati navade in obnašanje svojih strank, zato je sledenje spletnim uporabnikom postalo nujen del spleta. Sledenje omogoča dostop do statističnih podatkov o obiskanosti spletnih strani, odkrivanje vzorcev v obnašanju spletnih uporabnikov in merjenje uspešnosti spletnih oglaševalskih kampanj.

V diplomskem delu smo proučevali tehnike sledenja spletnim uporabnikom. Podrobno smo proučili napredno metodo sledenja spletnim uporabnikom z zajemom prstnega odtisa naprave oziroma brskalnika (angl. *device/browser fingerprinting*). Zanimala nas je predvsem uspešnost in zanesljivost tovrstne tehnike sledenja. Na podlagi preteklih raziskav na tem področju smo razvili lasten sistem prepoznave spletnih uporabnikov s pomočjo zajema prstnega odtisa naprave. V okviru diplomskega dela smo izvedli raziskavo, s katero smo analizirali delovanje našega sistema ter predlagali smernice za nadaljnje delo. Dotaknili smo se tudi pravnih in etičnih vidikov tovrstnega sledenja ter predstavili primere dobre in slabe uporabe tovrstne tehnologije.

Poglavje 2

Sledenje spletnim uporabnikom

Vse metode sledenja, tako v realnem kot v digitalnem svetu, temeljijo na prepoznavi oziroma identifikaciji uporabnika. Za prepoznavo je potreben identifikator, s katerim je posamezen uporabnik enolično določen. Spletne metode sledenja morajo same ustvariti identifikatorje, s pomočjo katerih uporabniku sledijo prek različnih zahtevkov ali celo spletnih strani. Metode se razlikujejo v načinu stvaritve identifikatorjev, prav tako pa je je različna hramba le-teh. V nadaljevanju bomo na kratko predstavili poznane metode sledenja in njihove razlike.

2.1 Klasične metode sledenja

V začetni fazi spleta so bile spletne strani precej enostavne, zato potrebe po prepoznavi uporabnikov še ni bilo. S popularizacijo in množično uporabo spleta pa se je vse več programerjev srečevalo s problemom, kako naj spletna stran oziroma spletni strežnik ohrani stanje obiskovalca skozi čas njegovega obiska. Protokol HTTP namreč ne omogoča upravljanja stanja povezav, zato tudi nima mehanizma, s katerim bi lahko prepoznal različne zahtevke istega obiskovalca. V začetni fazi je za identifikator zadostoval naslov IP. Z množično uporabo spleta pa je izgubil zanesljivost, saj je število spletnih uporabnikov preseglo število možnih naslovov IP, pričela pa se je uporabljati tehnologija NAT (angl. *Network Address Translation*), ki več napravam omogoča povezavo v splet prek enega naslova IP. Dinamično dodeljevanje naslovov IP in porast uporabnikov prenosnih naprav je identifikacijo na podlagi naslova IP še otežilo. Za prepoznavo medsebojno povezanih zahtevkov določenega

uporabnika iz množice navidezno neodvisnih zahtevkov se je zato začel uporabljati mehanizem skritega vnosnega polja, preko katerega se je prenašal identifikacijski niz. Hkrati se je pojavil alternativni mehanizem, kjer se je vsem zahtevkom, k naslovu URL, pripel identifikacijski niz uporabnika. Tako je strežnik lahko povezal zahtevke istega uporabnika in vodil njegovo sejo oziroma stanje. Obe rešitvi pa sta zahtevali precej veščega programerja in sta bili težavni za razhroščevanje.

Leta 1994 je programer Lou Montulli, tedaj zaposlen pri Netscape Communications, razvil rešitev za upravljanje seje, imenovano čarobni piškotek (angl. *magic cookie*). Rešitev je razvil za potrebe spletne košarice, ki jo je razvijal [4]. Njegova rešitev je bila kasneje formalizirana s strani skupine IETF v dokumentu RFC2109 leta 1997 [2], popravljena pa v dokumentu RFC2965 leta 2000 [3]. Najnovejša in trenutno aktualna različica je formalizirana v dokumentu RFC6265 [1] in še danes velja kot *de facto* standard za vodenje uporabniške seje.

Piškotek je majhna neizvršljiva tekstovna datoteka, velika praviloma do 4 kilobajte. Spletni strežnik ustvari piškotek in ga pošlje brskalniku, ta pa ga shrani v shrambo piškotkov. Brskalnik nato ob vsakem naslednjem obisku istega strežnika zahtevkom doda tudi piškotek. Piškotki so v osnovi pari *ključ - vrednost*. S pomočjo teh parov lahko strežnik identificira istega uporabnika preko različnih zahtev HTTP, kar omogoča vodenje seje uporabnika. Piškotki lahko poleg sejnih ključev shranjujejo tudi tudi druge poljubne pare, ki služijo različnim namenom, kot so nastavitev priljubljenega jezika, v katerem naj se spletna stran prikaže, prijavni podatki s katerimi se uporabnik samodejno vpiše v prijavni sistem spletne strani ali pa sledilni ključ preko katerega spletna stran prepozna obiskovalca, ki je stran predhodno že obiskal. Takšnim piškotkom pravimo trajni piškotki (angl. *persistent cookies*), saj omogočajo trajnost podatkov, do datuma, ki je določen ob kreiranju piškotka.

S prihodom HTML5 se je kot alternativa piškotkom pojavila spletna shramba (angl. *Web Storage*). Spletna shramba, tako kot piškotki, shranjuje pare *ključ - vrednost*. Prednost te tehnologije je predvsem hitrejša, bolj varna in enostavnejša uporaba. Vsebina spletne shrambe se ne pošilja ob vsaki zahtevi, temveč le ob izrecni zahtevi odjemalca. Drugačen je tudi način dostopa do podatkov, saj je mogoč le z odjemalčeve strani in ne strežniške. Količina podatkov je omejena na 5 megabajtov na posamezno domeno, kar omogoča shranjevanje kompleksnejših

podatkov. Podatki v lokalni shrambi nimajo določenega časa veljavnosti, saj z njimi, po priporočilu konzorcija W3C, upravlja uporabnik [5]. Brskalnik jih hrani, dokler jih uporabnik ali spletna aplikacija, ki jih je ustvarila, ne izbriše.

Posebna metoda sledenja so spletni podtaknjenci (angl. *Web Beacons*, *Web bugs* ali *Tracking bugs*), ki so kombinacija običajnih piškotkov, izvršljivih datotek in zahtev HTTP. Spletni podtaknjenec je skrit objekt, ki je lahko vgrajen v spletno stran, e-poštno sporočilo ali podpis uporabnika spletnega foruma. Z njihovo pomočjo je moč slediti uporabnikom spletnih strani in e-poštnih sporočil. Spletni podtaknjenci obstajajo v mnogih različnih oblikah, najbolj razširjeni pa so v obliki majhnih slik velikosti ene slikovne točke (1px). Vgrajeni so kot legitimne komponente spletne strani ali sporočila, uporabniku pa so večinoma popolnoma nevidni. Glavna značilnost podtaknjencev je, da niso del obiskane spletne strani ali e-poštnega sporočila, temveč so vir, ki ga mora brskalnik z zahtevkom pridobiti z druge spletne strani. Ker gre za običajen zahtevek HTTP, lahko druga stran te zahtevke beleži in jim sledi. Če je uporabnik to drugo spletno stran že kdaj obiskal in ima nastavljen sledilni piškotek, ga tako skupaj z zahtevo HTTP tudi pošlje.

Primeri najbolj razširjenih spletnih podtaknjencev so izvršljive datoteke gumbov socialnih omrežij (*Facebook Like sdk.js*, *Google +1 plusone.js* in podobni), spletna analitika (kot je *Google ga.js* in podobni) in oglasi spletnih oglaševalskih agencij. Ti podtaknjenci ponudnikom omogočajo sledenje spletnim uporabnikom preko velikega števila spletnih strani, tudi tistih, ki niso v njihovi domeni ali lasti.

Podtaknjenci so razširjeni tudi v e-pošti. S pomočjo podtaknjencev lahko beležimo število odprtih e-poštnih sporočil, ter kdaj in kdo ga je odprl. Storitve, kot je MailChimp, omogočajo vodenje e-poštnih kampanj, s pomočjo katerih lahko precej natančno spremljamo učinkovitost teh kampanj [9]. Delujejo s pomočjo podtaknjencev, ki vsebujejo identifikatorje kampanje ter naročnika, prijavljenega na določen e-poštni seznam.

Kot smo že omenili, so podtaknjenci lahko tudi izvršljive datoteke JavaScript, ki v ozadju pošiljajo podatke tretjim osebam. V spletnem članku [10] raziskovalec P. Eckersley opisuje raziskavo, ki jo je opravil leta 2009, kjer je odkril kar 10 različnih programov JavaScript s strani tretjih oseb, ki so delovali kot spletni podtaknjenci na popularni spletni strani *CareerBuilder.com*. Med njimi so bila tudi podjetja, kot so Google, AOL in Microsoft. Oglaševalske agencije se meha-

nizma podtaknjencev poslužujejo za namene analitike (merjenje števila prikazanih oglasov) ter grajenja profilov uporabnikov.

2.2 Napredne metode sledenja

Poleg klasičnih piškotkov poznamo tudi piškotke Flash in Silverlight. Delovanje je podobno delovanju običajnih piškotkov HTTP, le da piškotki Flash in Silverlight delujejo s pomočjo vtičnikov. Oboji omogočajo hrambo majhne količine podatkov za ohranjanje stanja, hranijo pa se na uporabnikovi napravi. Do istega piškotka Flash ali Silverlight imajo dostop vsi brskalniki, ki so naloženi na uporabnikovi napravi in imajo omogočen vtičnik (angl. *cross-browser*). Večina uporabnikov teh piškotkov ne pozna, saj so precej neopazni in skriti. Zaradi svojih lastnosti so piškotki Flash in Silverlight postali zanimivi tudi za sledenje spletnim uporabnikom.

Leta 2009 je skupina raziskovalcev z univerze Berkley ugotovila, da je več kot 50 od 100 svetovno najpopularnejših spletnih strani uporabljalo piškotke Flash. Med njimi je bilo tudi nekaj vladnih spletnih strani, kot je na primer *whitehouse.gov*, in strani večjih podjetij, med njimi AOL, Microsoft in Walt Disney [6]. V kasnejši raziskavi [7] so odkrili še eno zanimivost, piškotki so se na nekaterih straneh obnovili, čeprav so jih pred tem odstranili. To so bili *zombi* piškotki, ki so s pomočjo piškotkov Flash uspeli obnoviti običajne piškotke, ki so jih uporabniki izbrisali. *Zombi* piškotki so opisani v nadaljevanju.

Januarja 2011 so podjetja Google, Mozilla Fundation in Adobe Systems razvila vmesnik, ki je bil vključen v različico vtičnika Flash Player 10.3 in je omogočil, da se piškotki Flash obravnavajo enako kot običajni. Tako so brskalniki in s tem uporabniki dobili boljši nadzor nad uporabo piškotkov Flash, saj jih je moč izbrisati enako kot običajne piškotke. Podobne rešitve za piškotke Silverlight še ni.

Zombi piškotek pravzaprav ni en sam piškotek, temveč množica različnih piškotkov in drugih metod, ki omogočajo identifikacijo uporabnika. *Zombi* piškotek je lahko sestavljen iz:

- običajnih piškotkov HTTP,
- skupnih lokalnih objektov (piškotkov Flash),

- piškotkov Silverlight,
- piškotkov, shranjenih v RGB vrednostih slike PNG,
- piškotkov, shranjenih v ETag,
- piškotkov, shranjenih v brskalnikovem predpomnilniku,
- lokalne shrambe HTML5,
- sejne shrambe HTML5,
- globalne shrambe HTML5,
- piškotkov, shranjenih v podatkovni bazi SQLite.

Če uporabnik katerega od piškotkov izbriše, poskuša *zombi* piškotek obnoviti vrednosti iz ostalih piškotkov. Dovolj je, da se ohrani le en piškotek, z njim pa nato *zombi* piškotek obnovi ostale. Piškotki te vrste grobo kršijo zasebnost uporabnika na spletu, saj specifikacija RFC6265 [1] predpisuje transparentno uporabo piškotkov z možnostjo izbrisa s strani uporabnika. *Zombi* piškotki pa kljub jasno izraženi želji uporabnika po ne-sledenju, obnovijo sledilne piškotke. Tovrstne tehnike so uporabljala tudi večja podjetja kot so AOL, Hulu, Sears in Lowe's [6, 7]. Veliko pozornosti je vzbudila tudi objava prostodostopnega programa in razlage delovanja *zombi* piškotka, imenovanega *EverCookie* [8]. Spletna stran *http://samy.pl/evercookie/* prikazuje delovanje zombi piškotka, katerega izbris je zelo zahteven.

2.3 Sledenje z zajemom prstnega odtisa naprave in brskalnika

Ko uporabnik brska po spletu, brskalnik pošilja kopico zahtevkov spletnim stranem. Skupaj z zahtevki pošilja tudi množico meta-informacij. Brskalnik, skupaj z zahtevki, spletnim stranem pošilja informacije o različnih vrstah podatkov, ki jih zna prebrati, na primer HTML, XHTML, XML, GZIP, videe Flash, dokumente PDF in še mnogo drugih. Prav tako lahko pove, kateri jezik uporablja obiskovalec,

v katerem časovnem pasu je in kako velik zaslon ima. Za to obstaja več legitimnih razlogov. Nekateri podatki zagotavljajo hitrejše delovanje (GZIP), nekatere omogočajo pravilen prikaz podatkov (časovni pas), spet drugi omogočajo boljšo uporabniško izkušnjo (velikost zaslona). Svetovni splet je postal globalna vas in le redko kateremu Evropejcu bi podatek o nogometni tekmi, zapisan v brazilskem časovnem pasu, koristil. Spletne strani želijo uporabniško izkušnjo in vsebino vse bolj prilagoditi potrebam obiskovalcem. Temu seveda sledijo tudi brskalniki, ki spletnim stranem "izdajajo" vse več informacij, na podlagi katerih lahko spletne strani svojo vsebino prilagajajo. Porast mobilnih naprav je ta trend še pospešila.

Zajem prstnega odtisa naprave (angl. *device fingerprinting*) je metoda, ki tovrstne podatke izkorišča za potrebe identifikacije uporabnikov. Nekatere podatke je mogoče zajeti brez posebnega truda, saj jih brskalniki pošljejo sami in so ponavadi nujno potrebni za pravilno delovanje spleta. Takšen primer so glave zahtevkov HTTP, ki vsebujejo mnogo meta-informacij. Temu pravimo **pasivni zajem**. Več podatkov pa je mogoče pridobiti z **aktivnim zajemom**, kjer uporabnikovo napravo s pomočjo različnih metod sprašujemo po vrednostih določenih parametrov.

Vsaka naprava ima svoje značilnosti in lastnosti, s katerimi jo lahko opišemo. Na strojnem nivoju so primeri teh lastnosti velikost, ločljivost in tip zaslona, velikost pomnilnika, velikost diska, tip grafične kartice, prisotnost kamere ter mikrofona, hitrost in število CPE in še bi lahko naštevali. Na nivoju programske opreme so te lastnosti različica operacijskega sistema, različice naloženih aplikacij, gonilnikov in podobno.

Vsak uporabnik lahko računalnik prilagodi svojim potrebam. Prilagodi lahko določene nastavitve operacijskega sistema, kot sta jezik in časovni pas, ter naloži aplikacije, ki jih potrebuje. S tem postaja množica lastnosti in nastavitev uporabnikovega računalnika vse bolj unikatna. Ideja zajema prstnega odtisa naprave temelji ravno na predpostavki, da lahko dovolj velika množica lastnosti in nastavitev napravo enolično določa.

2.3.1 Teoretično ozadje

Ko se vprašamo, če lahko določen podatek o osebi to osebo tudi identificira, odgovor ni vedno preprost. Podatek moramo vedno postaviti v kontekst, če želimo ugotoviti njegovo pravo vrednost. Tako na primer podatek o dnevu rojstva osebe

ne more identificirati osebe v kontekstu države Slovenije, zlahka pa identificira učenca točno določenega razreda v osnovni šoli. Vidimo, da je kontekst pravzaprav tudi podatek. Če poleg datuma rojstva poznamo tudi spol osebe, še vedno nimamo dovolj informacije za identifikacijo. Če pa temu dodamo še podatek o kraju, v katerem oseba živi, je v mnogo primerih osebo mogoče identificirati. V navidezno anonimnih podatkih, kot so spol, datum rojstva in kraj osebe se namreč skriva določena količina informacije, ki je lahko dovolj velika za natančno identifikacijo. V študiji iz leta 2001 je profesor L. Sweeney ugotovil, da lahko samo na podlagi podatkov o spolu, datumu rojstva in poštni številki osebe identificira kar 87% oseb v ZDA [13]. To trditev prikazujemo v nadaljevanju na primeru.

Primer. Matematika pozna količino, ki ji pravimo entropija oziroma mera nedoločenosti naključne spremenljivke. Z njo lahko merimo količino informacije, ki jo vsebuje neka spremenljivka. Enota entropije je bit, izračunamo pa jo s pomočjo splošne enačbe (2.1).

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_b p(x_i) \tag{2.1}$$

Za primer vzemimo met kovanca. Kovanec je analogija bitu, oba imata le dve možni stanji, 0 in 1, grb in cifro. V primeru, ko je kovanec pošten, sta oba možna izida enako verjetna. Mera nedoločenosti je tedaj največja možna (*H(X) = 1*), napoved izida pa čista špekulacija. Ko pošten kovanec po metu pogledamo, pridobimo natanko 1 bit informacije, saj sta možna izida le dva. V primeru, ko so možni izidi štirje, je količina entropije 2 bita, pri osmih možnih izdih pa 3 biti. Pri metu poštene kocke, kjer je možnih izidov 6, po metu pridobimo $H(X) = 2,585$ bita informacije. Nasprotno pa se zgodi, če imamo kovanec, ki ima na obeh straneh grb. Vnaprej lahko z gotovostjo trdimo, da bo padel grb. Po metu kovanca res pade grb, s poznavanjem rezultata pa ne pridobimo nobene nove informacije, saj smo izid poznali že vnaprej. Entropija je v tem primeru enaka 0.

Vrnimo se na primer identifikacije oseb. Na svetu je trenutno okoli 7,2 milijarde ljudi. Če si izberemo naključnega zemljana, vsebuje njegova identiteta nekaj manj kot 33 bitov informacije (2.2).

$$H(X) = -\log_2(\frac{1}{7.200.000.000}) = 32,745 \tag{2.2}$$

Iz tega sledi, da za uspešno identifikacijo zemljana potrebujemo vsaj 32,745 bitov informacije oziroma 33 bitov ob bolj konzervativni oceni.

V spletnem zapisu [14] je P. Eckersley prikazal primer identifikacije zemljana. Ta primer bomo postavili v kontekst Slovenije. Pokazali bomo, kako lahko le s spolom, datumom rojstva in krajem bivanja identificiramo osebo.

Ob poenostavitvi, da so rojstni dnevi približno enako porazdeljeni preko celega leta, ugotovimo, da vsebuje podatek o datumu rojstva 8,51 bitov informacije (2.3).

$$H(X) = -\log_2(\frac{1}{365}) = 8,51 \tag{2.3}$$

Drugi podatek pravi, da oseba živi v najmanjši slovenski občini Hodoš/Hodos. Količina informacije, vsebovane v tem podatku, je 24,24 bitov (2.4).

$$H(X) = -\log_2(\frac{362}{7.200.000.000}) = 24,24 \tag{2.4}$$

Če torej seštejemo dobljeni količini informacije, dobimo 32,75 bitov. To bi morda že zadoščalo, vseeno pa dodajmo še informacijo o spolu. Za poenostavitev, bomo zopet predpostavili, da je na svetu enako razmerje moških in žensk. To pomeni, da podatek o spolu vsebuje 1 bit informacije (2.5). Skupna informacija treh podatkov torej vsebuje 33,75 bitov, kar je zelo verjetno dovolj za identifikacijo osebe.

$$H(X) = -\log_2(\frac{1}{2}) = 1 \tag{2.5}$$

Seveda je zgornji primer precej poenostavljen, vendar pa vseeno zelo zgovoren in prikazuje teoretično osnovo, na kateri temelji sledenje uporabnikov z zajemom podatkov.

Vzporednice lahko potegnemo v svet spleta. Naprave in brskalniki spletnim stranem posredujejo informacije, ki same po sebi še ne izdajo identitete uporabnikov. Kombinacija teh podatkov pa lahko vsebuje dovolj veliko količino informacije za enolično prepoznavo uporabnikove naprave.

2.3.2 Pretekle raziskave

Na področju zajema prstnega odtisa naprav za namen sledenja spletnim uporabnikom je bilo narejenih kar nekaj raziskav. Najbolj obširna raziskava je bila narejena leta 2010 s strani organizacije *Electronic Frontier Foundation* in avtorja P. Eckersleya [12]. V raziskavi so zajeli kar 470.161 prstnih odtisov brskalnikov s pomočjo spletne strani `http://panopticlick.eff.org`. Ugotovili so, da povprečen prstni

odtis vsebuje kar 18,1 bitov informacije. Kar 83,6% brskalnikov je ustvarilo unikaten prstni odtis, 5,3% pa si je prstni odtis delilo le še z enim brskalnikom. Pri uporabnikih, ki so imeli vključen vtičnik Flash ali Java Virtual Machine, je odstotek unikatnih prstnih odtisov dosegel 94,2%, 4,8% pa si je odtis delilo še z enim brskalnikom. Le en odstotek brskalnikov si je prstni odtis delilo z več kot enim brskalnikom. V raziskavi so ugotovili, da je stabilnost prstnega odtisa relativno majhna, saj se odtis brskalnika s časom hitro spreminja. To pomanjkljivost so uspešno minimizirali s preprostim algoritmom, ki je v kar 99,1% primerov spremembo pravilno zaznal in jo pripisal pravemu brskalniku.

Podobni raziskavi sta objavila tudi H. Tillman in R. Broenink [15, 16]. V raziskavi, ki je trajala dva meseca, je Tillman zajel 23.709 prstnih odtisov brskalnikov. Ugotovil je, da je kar 93,6% odtisov unikatnih, pri čemer je bilo 89,9% prstnih odtisov brskalnikov sorazmerno stabilnih in se s časom niso močno spreminjali. Broenink je v svoji raziskavi zbral 1124 prstnih odtisov brskalnikov, od katerih je bilo 95,9% odtisov brskalnikov unikatnih. S preprostim algoritmom je kasneje uspešno prepoznal 86,6% vračajočih se uporabnikov, ki se jim je prstni odtis brskalnika zaradi različnih posodobitev spremenil.

Malce drugačen pristop so ubrali raziskovalci K. Boda in sod. [11]. Raziskovali so zajem prstnih odtisov naprave, ki je neodvisen od brskalnika (angl. *cross-browser*). V članku opisujejo metodo, ki le s podatki, kot so naslov IP, prisotnost določenih pisav, časovni pas in ločljivost zaslona, uspešno identificirajo večino uporabnikov. Zbrali so testno množico skoraj tisoč prstnih odtisov. Za razliko od predhodnih raziskav so uvedli novo metodo zajema seznama pisav. Namesto uporabe vtičnika Flash, ki vrne celoten seznam pisav, so se odločili za zajem seznama s pomočjo programa JavaScript, ki preverja prisotnost posamezne pisave. Tej odločitvi je botrovalo dejstvo, da okoli 10% uporabnikov ne uporablja vtičnika Flash, kar drastično zmanjša možno količino zajete informacije. V članku na žalost eksplicitno ne omenjajo uspešnosti prepoznave njihovega sistema, opisujejo pa analizo zbranih podatkov in učinkovitost posameznih identifikatorjev. Ugotovili so, da so se prstni odtisi naprav porazdelili v relativno majhne anonimne podmnožice velikosti manj kot 10 naprav. Anonimne podmnožice so množice, v katerih so naprave z enakimi prstnimi odtisi. Množice z le enim elementom niso več anonimne in omogočajo identifikacijo naprave. Njihov sistem prepoznave smo

si izbrali za izhodišče našega sistema prepoznave.

Dve zanimivi študiji o zajemu prstnih odtisov naprav je objavil tudi K. Mowery s sodelavci. V prvi študiji [17] so proučili metodo, ki meri čas izvajanja določenih temeljnih funkcij jezika JavaScript. Metoda ne išče funkcionalnih razlik med brskalniki, temveč razlike v zmogljivosti. S tem ne zazna le razlik med brskalniki, temveč tudi nekatere mikro-arhitekturne značilnosti, ki navadno niso opazne. Druga metoda [18] pa za identifikacijo izkorišča delovanje elementa *canvas*, ki je novost v HTML5. Element *canvas* je površina, na katero lahko uporabnik ali program JavaScript riše. Za upodabljanje (angl. *rendering*) slike ponavadi skrbi grafična kartica naprave, zato je rezultat upodabljanja odvisen predvsem od nje. Rezultat je, poleg grafične kartice, odvisen tudi od gonilnikov, ki upravljajo z grafično kartico. Raziskovalci so odkrili, da so rezultati pri upodabljanju enake slike na različnih grafičnih karticah in gonilnikih lahko različni. Prostemu očesu razlika morda ni opazna, če pa med seboj primerjamo korespondenčne slikovne točke (angl. pixels), so razlike očitne. V omenjeni raziskavi so prikazali, da lahko z uporabo tovrstne metode identificirajo posamezne grafične kartice ter razlikujejo računalnike med seboj. Omenjena metoda sicer ni dovolj za uspešno identifikacijo posameznih naprav, lahko pa služi kot dober in stabilen parameter prepoznave.

Enega izmed najbolj zanimivih pristopov pa so predstavili raziskovalci Kohno, Broido in Claffy v članku [19]. Predstavili so namreč metodo zajema prstnega odtisa naprave, ki temelji na mikroskopskih razlikah v odmiku ure (angl. *clock-skew*). Skoraj vsi današnji računalniki za merjenje realnega časa uporabljajo integrirano vezje z uro RTC (angl. *Real-Time Clock*), ki deluje s pomočjo nihanja kristala. Ti kristali imajo svoje omejitve glede natančnosti. Frekvenca nihanja je običajno 2^{15} nihajev na sekundo, vendar pa lahko nekateri dejavniki, kot sta temperatura in fizična zgradba kristala vplivajo na frekvenco nihanja. To pomeni, da nobena ura RTC nima povsem enake frekvence nihanja. Vsaka ura RTC ima namreč nek zelo majhen in konstanten odmik, zaradi česar moramo sistemsko uro vsakih nekaj dni sinhronizirati. V članku raziskovalci opisujejo metodo, ki zna ta majhen odmik ure izmeriti ter ga uporabiti kot identifikator naprave. Metoda deluje le na sistemih Unix, zato ni primerna za splošno identifikacijo.

Raziskovalci z univerze KU Leuven so konec leta 2013 raziskovali razširjenost metode sledenja uporabnikom z zajemom prstnega odtisa naprave [20]. Prečesali

so prvih 10.000 globalno najbolj popularnih spletnih strani in odkrili 145 (1,5%) spletnih strani, ki so uporabljale znane metode zajema prstnih odtisov naprav. Ugotovili so tudi, da se pojavlja vse več podjetij, ki ponuja plačljive programske rešitve za zajem prstnih odtisov naprav.

2.4 Zakonodaja in zaščita

V tem poglavju bomo predstavili nekatere pravne vidike uporabe piškotkov in drugih tehnik sledenja spletnim uporabnikov. Ker je tema precej obširna, se je bomo le bežno dotaknili in izpostavili bistvene aspekte zakonodaje po svetu in pri nas. V nadaljevanju bomo pregledali tudi tehnike za zaščito pred sledenjem.

2.4.1 Zakonodaja

Na globalni ravni še ni konsenza glede zakonske regulacije sledenja spletnim uporabnikom. Države same urejajo zakonodajo na področju zasebnosti in varovanja osebnih podatkov, zaradi česar prihaja do pomembnih razlik med zakonodajami različnih držav. Internet pa za razliko od držav ne pozna meja, zato prihaja do precejšne zmede in nespoštovanja pravnih ureditev.

Kot pojasnjuje Prastalo [21], je zanimivo, da prav ZDA, kot vodilna svetovna in spletna velesila, področje ureja zelo pomanjkljivo. Ustava ZDA nikjer eksplicitno ne omenja varovanja zasebnosti, vendar pa je z interpretacijo določenih amandmajev mogoče zaključiti, da ustava vseeno varuje določene aspekte zasebnosti. Kot opisujeta Baumer in Solove, ima neodvisna ameriška agencija FTC (Federal Trade Commision) osrednjo vlogo na področju zasebnosti na spletu. Od spletnih strani ne zahteva objave politike zasebnosti (angl. *privacy policy*), jo pa priporoča. Če se spletna stran odloči, da politiko zasebnost objavi, jo mora tudi spoštovati. V nasprotnem primeru jo doletijo sankcije agencije FTC [22, 23]. Politiko zasebnosti si torej določajo spletne strani same, dolžnost uporabnikov pa je, da si jo preberejo in odločijo, ali se z njo strinjajo ali ne. Ker je politika zasebnosti ponavadi dolgo in zapleteno besedilo, jo uporabniki le redko preberejo. Leta 2011 je predstavnica spodnjega doma kongresa, Jackie Speier, predstavila predlog zakona, poimenovanega *Do Not Track Me Online Act 2011*. V predlogu naj bi se sledenje spletnim uporabnikom zakonsko precej omejilo. Uporabnikom pa bi se morala omogočiti

tudi izbira, da sledenje globalno prepovejo preko nastavitev brskalnika, točneje preko nastavitve DNT (DoNotTrack). Zakon sicer ni bil nikoli potrjen, so pa zato vsi glavni proizvajalci brskalnikov sčasoma uvedli mehanizem DNT v svoje brskalnike [24]. Ker predlog zakona ni bil sprejet, spletne strani mehanizma DNT zakonsko niso dolžne spoštovati.

V Evropski uniji je poudarek na zagotavljanju zasebnosti na spletu mnogo večji kot v ZDA. Direktiva o varovanju podatkov (95/46/EC) [25] je bila sprejeta že leta 1995. Direktiva pokriva področje varovanja in dela z osebnimi podatki, ni pa se direktno opredeljevala do podatkov na spletu, kot pravi Prastalo [21]. Po razmahu interneta je EU leta 2002 sprejela direktivo o zasebnosti na spletu (2002/58/EC) [26], ki obravnava področja zaupnosti podatkov, obdelave podatkov o prometu, nezaželene komunikacije (angl. *spam*) in sledenja spletnim uporabnikom. Leta 2009 je bila dopolnjena z direktivo o zasebnosti in elektronskih komunikacijah (2009/136/EC) [27], ki je uvedla nekaj pomembnih sprememb na področju varovanja zasebnosti na spletu [22]. Direktiva ureja tudi področje piškotkov in drugih tehnik sledenja, kjer predpisuje, da mora imeti uporabnik možnost vpogleda in kontrole nad sledenjem, uporaba tehnologij sledenja pa je mogoča le na podlagi privolitve uporabnika, pojasnjuje Baumer [22].

Države članice, med njimi tudi Slovenija, so dolžne spoštovati in izvajati odločbe direktive in jih uvesti v svoj pravni red. Slovenija je na podlagi direktive spremenila Zakon o elektronskih komunikacijah (ZEKom-1) [28]. Začel je veljati v začetku leta 2013, prinesel pa je nova pravila glede uporabe piškotkov in podobnih tehnologij za shranjevanje informacij ali dostop do informacij, shranjenih na računalniku ali mobilni napravi uporabnika. Kot pojasnjuje Informacijska pooblaščenka Republike Slovenije, nova zakonodaja uporabe piškotkov ne prepoveduje, pač pa zaostruje pravila – predpisuje, pod kakšnimi pogoji in kako je piškotke in podobne tehnologije dovoljeno uporabljati. Uporabniki spletnih strani morajo biti s piškotki seznanjeni, ponujena pa jim mora biti izbira, ali želijo, da spletna stran na tak način spremlja njihove aktivnosti na spletu [29]. Ključno dejstvo pri direktivi je, da piškotkov direktno niti ne omenja, temveč se osredotoča predvsem na rezultat uporabe orodij, kot so piškotki. Direktiva je v tem smislu namreč tehnološko nevtralna [30].

Zakonodaja je zelo jasna tudi v primeru zajema podatkov o napravi za namene

sledenja uporabnikom (*device/browser fingerprinting*). To v smernicah pojasnjuje tudi Informacijska pooblaščenka:

> "Zakon se nanaša na uporabo piškotkov in vseh drugih tehnologij, ki omogočajo shranjevanje podatkov ali pridobivanje dostopa do podatkov, shranjenih v terminalski opremi naročnika (kot npr. t.i. piškotki Flash oz. Local Sharared Objects, Web beacon ali bug oz. clear gif). Vse več se uporablja tudi prepoznava podpisa uporabnikove naprave oz. brskalnika (device/browser fingerprinting), ki sicer ne uporablja piškotkov, a lahko prepozna uporabnika. Tudi v slednjem primeru pridejo v poštev pravila o varovanju osebnih podatkov ter tako lahko tudi določbe nove zakonodaje." [29]

Podobno se do različnih oblik sledenja opredeljujeta tudi britanski informacijski komisar (ICO) [32] in evropska delovna skupina za zaščito podatkov (*Article 29 Working Party*) [31]. Pomembno je tudi poudariti, da se lahko prstni odtis naprave smatra kot osebni podatek in je zato lahko podvržen zakonodaji o varovanju osebnih podatkov [29, 31, 32].

Na področju Evropske unije zakonodaja torej ščiti zasebnost uporabnikov in se natančno opredeljuje do sledenja in zbiranja podatkov o uporabnikih. Problematično pa je, da globalno usklajene zakonodaje na tem področju ni. Internet državnih mej namreč ne pozna, zato so uporabniki zunaj EU prepuščeni lokalnim zakonom in lastni zaščiti.

2.4.2 Zaščita posameznika

V tem poglavju bomo na kratko predstavili možnosti, ki jih imajo uporabniki za zaščito pred sledenjem. Na spletu je dostopnih veliko orodij, ki dvigujejo raven zasebnosti uporabnikov, vsi pa imajo tako svoje prednosti kot slabosti.

Zaščita pred klasičnim načinom sledenja

Klasičen način sledenja zajema metode, ki smo jih opisali v razdelku 2.1. Za zaščito pred tovrstnim sledenjem je priporočeno, da uporabniki v brskalniku blokirajo nalaganje piškotkov s strani tretjih oseb (angl. *3rd party cookies*). Blokiranje vseh piškotkov ni priporočeno, saj je pravilno delovanje mnogih strani mogoče le z

uporabo piškotkov. Višjo stopnjo zasebnosti je mogoče doseči tudi z nastavitvijo, da se vsi piškotki po zaprtju brskalnika izbrišejo. Žal se s tem zbrišejo tudi piškotki, ki omogočajo prijavo brez vsakokratnega vpisovanja gesel.

Višjo stopnjo zasebnosti ponujajo tudi nekatere razširitve brskalnikov. Razširitvi Ghostery in AdBlock na primer blokirata nalaganje vsebine in piškotkov s strani tretjih oseb, ter vse splošno znane metode sledenja. Uporabniku prikažeta, katere vsebine sta blokirali, omogočata pa mu tudi preprosto določanje izjem. Žal imata tudi svoje pomanjkljivosti, saj se zaradi njiju lahko zgodi, da se majhen delež spletnih strani ne prikaže pravilno.

Priporočena je tudi ročna kontrola uporabe vtičnikov. V nastavitvah brskalnika lahko uporabnik za vsak vtičnik posebej nastavi način zagona. Priporočeno je, da uporabnik za vsak vtičnik nastavi možnost, da ga brskalnik pred zagonom vtičnika vpraša za dovoljenje za zagon. Na ta način so funkcionalnosti minimalno okrnjene, uporabnik pa ima pregled nad uporabo vtičnikov. Mnogo metod sledenja namreč izkorišča samodejni zagon vtičnikov.

Zaščita pred zajemom prstnega odtisa naprave

Zaščita pred sledenjem z zajemom prstnega odtisa naprave je malce bolj zahtevna. Zajem prstnega odtisa naprave temelji namreč ravno na prepoznavanju drobnih razlik med brskalniki in njihovimi nastavitvami. V primeru, da bi se uporabnik želel maksimalno zaščititi, tako da bi blokiral vse piškotke, blokiral zagon vtičnikov in bi imel nameščene razširitve, ki blokirajo določene vsebine, se lahko zgodi, da se s tem še bolj izpostavi iz množice in postane bolj razpoznaven. Temu pravimo paradoks zajema prstnega odtisa (angl. *device fingerprinting paradox*), kot ga opisujeta Eckersley in Broenink [12, 16]. Problem tiči v dejstvu, da velika večina spletnih uporabnikov ne spreminja nastavitev svojega brskalnika. Poleg tega se veliko uporabnikov tudi ne zaveda razširjenosti sledenja, tako klasičnega kot sledenja z zajemom prstnega odtisa, in zato ne uporabljajo nikakršne zaščite. Zaradi tega uporabniki, ki zaščito uporabljajo, toliko bolj izstopajo iz množice.

Najbolj praktičen nasvet je, naj uporabnik ne izstopa iz množice. Uporaba najbolj pogostih nastavitev, najbolj pogostega brskalnika in operacijskega sistema namreč otežujeta identifikacijo uporabnika. Pri tem je pomembno poudariti, da velik del zajema prstnega odtisa temelji na seznamu pisav uporabnika. Nastavitev

ročnega vklopa vtičnika Flash je zato zelo priporočljiva, saj uporabnik s tem blokira zajem seznama pisav, ki je najbolj učinkovit identifikator. Ker pa je seznam pisav možno zajeti tudi s pomočjo programa JavaScript, je uporabnik pred trenutno težko rešljivim problemom. JavaScript je namreč nepogrešljiv del spletnih strani, zato je izklop zelo nepraktičen, uporabnik pa z izklopom tudi tvega, da bo tako postal bolj prepoznaven. Dobrih rešitev zaenkrat še ni. Zaradi tega je pomembno, da proizvajalci brskalnikov čim hitreje ponudijo rešitev, ki preprečuje zajem seznama pisav. Hitra in učinkovita rešitev bi bila že ta, da brskalniki preprečijo uporabo več kot 30 različnih pisav na spletni strani. S tem bi se zajem pisav zelo otežil, funkcionalnost spletnih strani pa bi se le redkokje zmanjšala. Pomemben korak so naredili že razvijalci vtičnika Flash, ki so v različici 14 onemogočili zajem pisav, če je uporabnik v načinu brez beleženja zgodovine. Prav tako je zelo pomembno, da se proizvajalci brskalnikov poenotijo in podatke, ki jih je možno pridobiti, posplošijo ter s tem zmanjšajo razlike med brskalniki. Že s skrčenjem niza UAS (angl. *User Agent String*) na le najbolj nujne informacije, bi se učinkovitost zajema drastično zmanjšala. Prav tako so nekateri brskalniki preveč zgovorni pri naštevanju vtičnikov ter tipov MIME. Pri naštevanju namreč izdajo preveč podatkov, ki sploh niso pomembni za delovanje. Tako na primer večina brskalnikov poleg imena vtičnika pove še natančno različico vtičnika, številko revizije, številko gradnje in sistemsko končnico datoteke!

Kot edini brskalnik, ki veliko pozornosti namenja tudi zaščiti pred zajemom prstnega odtisa naprave, se je izkazal Tor. Tor je že v svojem bistvu brskalnik, ki ščiti zasebnost uporabnikov, v naši raziskavi pa je bil edini, kjer je bil zajem podatkov zelo okrnjen.

Poglavje 3

Implementacija sistema sledenja z zajemom prstnega odtisa naprave

Ob proučevanju metod sledenja spletnim uporabnikom nas je najbolj zanimala metoda sledenja s pomočjo zajema prstnega odtisa naprave in brskalnika. Kot smo omenili že v poglavju 2.3, so bile nekatere metode na tem področju presenetljivo uspešne, zaščita pred njimi pa je pogosto jalova. Zato smo se odločili, da bomo razvili lasten sistem prepoznave spletnih uporabnikov ter preučili delovanje takšnega sistema z izvedbo eksperimenta na realni množici spletnih uporabnikov.

Cilj praktičnega dela diplomskega dela je razviti sistem, ki bo sposoben sledenja spletnim uporabnikom na podlagi prstnega odtisa njihovega brskalnika.

Zadali smo si naslednje ciljne lastnosti sistema:

- enoličnost prstnega odtisa, isti brskalnik mora vedno ustvariti enak prstni odtis,
- unikatnost prstnega odtisa, različni brskalniki ne smejo imeti enakih prstnih odtisov,
- robustnost, sistem mora delovati na vseh napravah,
- hitrost sistema,

- nevidnost, zajem mora biti uporabniku neopazen,
- odpornost na spremembe, sistem mora biti v zadostni meri odporen na spremembe in posodobitve uporabnikove naprave.

Poleg naštetih ciljev smo želeli preveriti tudi uspešnost prepoznave naprave (angl. *cross-browser fingerprinting*) brez parametrov brskalnika.

3.1 Izbira parametrov

Iz preteklih raziskav smo izluščili najbolj obetavne tehnike zajema podatkov, preverili pa smo tudi kopico parametrov, za katere smo menili, da lahko izboljšajo delovanje sistema prepoznave uporabnikov. Množica vseh možnih parametrov zajema je ogromna, zajamemo lahko več kot 100 različnih parametrov, kar lahko naredi sistem zajema precej zahteven. Poleg tega so določeni parametri neuporabni za naše potrebe, saj vsebujejo zelo malo ali pa sploh nič uporabne informacije. Nekateri parametri so redundantni, nekateri pa so precej odvisni od drugih parametrov. Zaradi tega smo morali analizirati posamezne parametre ter izbrati množico parametrov za zajem.

Za analizo parametrov smo izvedli raziskavo na realni množici 147 uporabnikov, od katerih smo pridobili vrednosti posameznih parametrov. Zbrane podatke smo analizirali, ter odstranili parametre, ki smo jih prepoznali kot nekoristne. Na podlagi analize smo se odločili za zajem nekaterih dodatnih parametrov, vsi izbrani parametri pa so opisani v nadaljevanju.

Izbrali smo malo manj kot 60 različnih parametrov, ki jih lahko razdelimo v 9 podskupin. Vsako skupino smo predstavili ter opisali način zajema. Ker je v nekaterih skupinah parametrov preveč, smo predstavili le najbolj pomembne iz vsake skupine. V tabelah so prikazani primeri dejanskih vrednosti parametrov, ki jih je vrnil brskalnik Google Chrome različice 35. Brskalnik je tekel na prenosnem računalniku z operacijskim sistemom Windows 7.

1 Glave zahtev HTTP

Prvo skupino parametrov sestavljajo glave zahtev HTTP, kot jih prikazuje tabela 3.1. Glave zahtev HTTP se prenesejo pri vzpostavitvi povezave brskalnika s sple-

tnim strežnikom. Zajamemo jih s strežniškim programom.

parameter	**vrednost**
Remote-Address (IP)	188.230.151.127
Connection	keep-alive
Referer	https://www.facebook.com/
X-Forwarded-For	null
Accept	text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
User-Agent	Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1985.125 Safari/537.36
DNT(DoNotTrack)	1
Accept-Encoding	gzip,deflate,sdch
Accept-Language	sl-SI,sl;q=0.8,en-GB;q=0.6,en;q=0.4

Tabela 3.1: Parametri glav zahtev HTTP.

2 Podatki o zaslonu

V drugo skupino parametrov spadajo podatki o zaslonu, kot jih prikazuje tabela 3.2. Vsak brskalnik omogoča dostop do objektov *screen* in *window*, ki vsebujeta podatke o dimenzijah in barvni globini zaslona, zajamemo pa jih lahko s pomočjo ukazov v okolju JavaScript. Objekta, poleg ločljivosti, vsebujeta tudi podatke o razpoložljivi ločljivosti, ki je na voljo brskalniku (*availWidth* in *availHeight*). S podatkom o razpoložljivi ločljivosti lahko zaznamo višino orodne vrstice operacijskega sistema. Opozoriti moramo tudi na dejstvo, da je parameter *devicePixelRatio* del objekta *window* in ne *screen*. Vsebuje podatek o faktorju povečave (angl. zoom). Večina mobilnih naprav prikazuje manjše ločljivosti zaslona od dejanske, imajo pa zato privzet faktor povečave večji od ena.

3 Objekt navigator in časovni odmik

Objekt *navigator* vsebuje podatka *navigator.language* in *navigator.platform*. V to skupino smo uvrstili še parameter, ki sicer ni del objekta *navigator*, je pa prav tako pomemben. To je parameter, ki vsebuje podatek o odmiku časovnega pasu (angl. *timezone offset*). Vse parametre zajamemo s pomočjo programa JavaScript. Primeri podatkov so prikazani v tabeli 3.3.

parameter	**vrednost**
screen.width x *screen.height*	1366 x 768
screen.availWidth x *screen.availHeight*	1366 x 728
screen.colorDepth	24
screen.pixelDepth	24
window.devicePixelRatio	1

Tabela 3.2: Parametri zaslona.

parameter	**vrednost**
navigator.language	sl-SI
navigator.platform	Win32
časovni pas (odmik od GMT)	120

Tabela 3.3: Objekt *navigator* in odmik časovnega pasu.

4 Podatki o napravi in brskalniku

V četrto skupino smo uvrstili parametre, ki vsebujejo podatke o napravi in brskalniku. Parametri so pravzaprav derivat niza *User-Agent*, ki vsebuje podatke o operacijskem sistemu, brskalniku in tipu naprave. Iz niza *User-Agent* lahko izluščimo parametre, ki so predstavljeni v tabeli 3.4. Podatki so izluščeni iz dejanskega niza: *Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.1916.153 Safari/537.36*. Za zajem teh parametrov smo uporabili prostodostopen program *WhichBrowser.js*, ki je dostopen na naslovu `https://github.com/NielsLeenheer/WhichBrowser` (16.7.2014).

5 Vtičniki

V skupino 5 smo uvrstili podatke o omogočenih vtičnikih. Ker je množica vseh mogočih vtičnikov ogromna, bi bil zajem vseh prezahteven. Podatki o vtičnikih se tudi hitro spreminjajo, saj se vtičniki pogosto posodabljajo. Zato smo se odločili, da bomo v zajem vključili le 6 najbolj pogostih vtičnikov, ki so prikazani v tabeli 3.5. Za pridobivanje seznama omogočenih vtičnikov smo uporabili prostodostopen program *pluginDetect.js*, ki je dostopen na naslovu `http://www.pinlady.net/`

parameter	**vrednost**
operacijski sistem	Windows 6.1
brskalnik	Chrome 35.0.1916.153
pogon brskalnika	Blink
arhitektura sistema	64 bitna
tip naprave	desktop

Tabela 3.4: Parametri izluščeni iz niza *User-Agent*.

`PluginDetect/` (16.7.2014). S pomočjo tega programa lahko zaznamo vključene vtičnike ter njihovo različico.

parameter	**vrednost**
Shockwave	Shockwave 11,6,5
Silverlight	Silverlight 5,1,30214,0
Java	Java 1,7,0,60
Adobe Acrobat Reader	AdobeReader 10,1
Flash Player	Flash 13,0,0
QuickTime	QuickTime 7,7,2

Tabela 3.5: Seznam naloženih vtičnikov.

6 Podatki vtičnika Silverlight

V šesto skupino smo uvrstili parametre, ki jih vsebuje vtičnik Silverlight. Podatke zajamemo s pomočjo programa, ki se izvaja v vtičniku Silverlight, za kar mora biti vtičnik seveda naložen in omogočen. Silverlight je zelo zgovoren, saj vsebuje nekaj zelo podrobnih podatkov o uporabnikovi napravi. Vsebuje podatek o številu jeder CPE, zelo podrobno različico operacijskega sistema in Silverlight okolja ter podatke o jezikovnih nastavitvah sistema. Primer je prikazan v tabeli 3.6. Za naše potrebe smo predelali prostodostopno različico programa DeviceInfo, ki je dostopen na naslovu `https://github.com/dimalinux/DeviceFingerprint/tree/master/src/main/silverlight` (10.7.2014).

parameter	**vrednost**
Environment.OSVersion.Version	6.1.7601.65536
Environment.OSVersion	Microsoft Windows NT 6.1.7601 Service Pack 1
Environment.Version	5.1.30214.0
ImageRuntimeVersion	v4.0.30319
Environment.ProcessorCount	2
CultureInfo.CurrentCulture	sl-SI
CultureInfo.CurrentUICulture	en-US

Tabela 3.6: Parametri vtičnika Silverlight.

7 Seznam nameščenih pisav Flash

V osmo skupino smo uvrstili seznam nameščenih pisav (angl. *fonts*) na uporabnikovi napravi. Seznam lahko zajamemo s programom, ki deluje v vtičniku Flash. Vključuje vse lokalno shranjene pisave, ki so vidne vtičniku Flash. Na tem mestu je pomembno pojasniti, kako se pisave shranjujejo na računalnik. Vsak operacijski sistem ima za sistemske pisave točno določen centralni imenik, v našem primeru je to *C:/Windows/Fonts*. Nekaj deset pisav se v imenik shrani skupaj z namestitvijo operacijskega sistema. Mnogo programov, ki jih namestimo na računalnik, prav tako shranjuje pisave v ta imenik. Najbolj pogosti primeri teh programov so *Microsoft Office*, *Adobe Photoshop*, *TeamViewer*, *Visual Studio*, *AutoCAD*, *Android SDK*, podobno pa deluje tudi vrsta drugih. Uporabnik lahko pisave v ta imenik shranjuje tudi sam. Tako lahko postane seznam pisav zelo raznolik, v mnogo primerih pa celo unikaten. V raziskavi parametrov je imelo od vseh uporabnikov, ki so imeli omogočen vtičnikom Flash, kar 91% uporabnikov popolnoma unikaten seznam. Seznam pisav je zaradi tega eden od najpomembnejših parametrov za zajem. Zanimivo je, da vtičnik Adobe Flash seznama pisav ne uredi po abecednem vrstnem redu, temveč vrne seznam, ki je urejen po času namestitve, kar ga naredi še bolj unikatnega.

8 Seznam nameščenih pisav JavaScript

V raziskavi parametrov smo ugotovili, da dobrih 10% uporabnikov namiznih računalnikov nima omogočenega vtičnika Flash, zato pri njih metoda zajema seznama

pisav z vtičnikom Flash ni delovala. Ker je ta parameter ključen za uspešno prepoznavo, smo morali poiskati bolj robustno metodo zajema seznama pisav. V raziskavi [11] so uporabili zanimiv pristop zajema seznama pisav z uporabo programa JavaScript. Proučili smo delovanje njihove metode zajema ter našli prostodostopen program *fontdetect.js* za zaznavo seznama pisav, ki je dostopen na naslovu `http://www.lalit.org/lab/javascript-css-font-detect/` (9.7.2014).

Ker je ta metoda zajema najbolj pomembna hkrati pa tudi najbolj kompleksna, zahteva nekaj obrazložitve. Metoda deluje po principu trkanja oziroma preizkušanja (angl. *knocking*). Vsak brskalnik ima namreč vgrajen mehanizem nadomestne pisave (angl. *fallback font*). S pravili CSS lahko programer določi primarno ter nadomestno pisavo besedila, ki se prikaže v primeru, ko primarna pisava ni na voljo. Primer pravila:

```
.div-class {
        font-family: Arial, monospace;
}
```

V zgornjem primeru poskusi brskalnik najprej besedilo zapisati s primarno pisavo *Arial*. Če uporabnik pisave nima naložene, se sproži mehanizem *fallback*, ki za prikaz besedila uporabi naslednjo, nadomestno pisavo. V zgornjem primeru je to pisava *monospace*, ki jo znajo pokazati vsi brskalniki. Poleg pisave *monospace* obstajata še pisavi *sans* in *sans-serif*, ki jih poznajo vsi brskalniki, v nadaljevanju pa jim bomo rekli temeljne pisave. S pomočjo tega mehanizma je mogoče preveriti prisotnost oziroma odsotnost posamezne pisave.

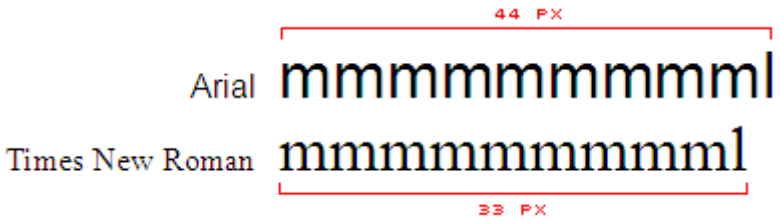
Preverjanje deluje tako, da si izberemo poljuben niz, ki ga najprej zapišemo s temeljno pisavo. S pomočjo programa JavaScript izmerimo višino in dolžino prikazanega besedila, vrednosti pa si shranimo za kasnejšo primerjavo. Isti niz nato zapišemo s pomočjo prej omenjenega pravila CSS. Za primarno pisavo določimo pisavo, katere prisotnost želimo preveriti, za nadomestno pisavo pa določimo temeljno. Ponovno izmerimo višino ter dolžino besedila in jo primerjamo z vrednostmi poprej izmerjene temeljne pisave. Če se vrednosti razlikujejo, je pisava prisotna, če pa so vrednosti enake, pisava ni na voljo, saj se je v obeh primerih besedilo prikazalo v temeljni pisavi. Razlike med pisavami so pogosto precej očitne, kot je vidno na sliki 3.1. V nekaterih primerih pa se lahko zgodi, da je pisava

zelo podobna temeljni. Zato program *fontdetect.js* vsako pisavo preveri s tremi različnimi temeljnimi pisavami, *monospace*, *sans* in *sans-serif*, kar naredi zaznavo precej zanesljivo.

Programu *fontdetect.js* moramo podati seznam pisav, ki jih želimo preveriti. Sprva smo poskusili program zagnati z vsemi pisavami (4801), ki smo jih zajeli v raziskavi parametrov, vendar se je algoritem izvajal več kot minuto. To je bilo z vidika praktičnosti popolnoma neuporabno, zato smo bili primorani seznam pisav zmanjšati. V podatkovni bazi smo analizirali in prešteli pojavitve posameznih pisav, poleg pa smo izračunali še deleže zastopanosti teh pisav v posameznih operacijskih sistemih. Tako smo lahko ugotovili določene zakonitosti pri nalaganju pisav. Izkazalo se je, da je mnogo pisav medsebojno odvisnih, saj se nalagajo v paketih s strani operacijskih sistemov in uporabniških programov. Zato smo iz vsakega zaznanega paketa ohranili le po eno pisavo, ostale pa odstranili. Delikatne pa so bile pisave, ki so bile zastopane v zelo majhnih številih, saj bi z njihovo odstranitvijo drastično zmanjšali varianco seznama pisav med uporabniki. Ravno splošno nerazširjene pisave namreč najbolj poudarijo razlike med napravami. V podatkovni bazi smo opazili uporabnika, ki je imel naloženih kar 2061 različnih pisav, med njimi pa je bilo precej takih, ki jih ni imel noben drug uporabnik. Zaradi tega je precej izstopal iz množice, kjer je imel povprečen uporabnik okoli 240 različnih pisav. Po podrobnem pregledu se je izkazalo, da je imel nameščen paket pisav, ki je prostodostopen na spletu. Zato smo na spletu poiskali podobne popularne pakete pisav in iz vsakega vključili par pisav v naš seznam. S celotno optimizacijo smo zmanjšali skupno število pisav na okoli 700. Z dodatnim testiranjem smo zaznali, da je bil v nekaterih primerih seznam zaznanih pisav na isti napravi v različnih brskalnikih drugačen. Tudi te pisave smo odstranili, končno število pisav pa se je tako zmanjšalo na 618. Čas izvajanja se je precej izboljšal, zmanjšal se je na vsega 1 do 10 sekund, odvisno od brskalnika in zmogljivosti naprave.

3.2 Izbira tehnologije

Sistem, ki smo ga razvijali, je v tehnološkem smislu zelo heterogen. Celoten pristop zajema prsnega odtisa naprave namreč temelji na precej raznoliki množici podat-



Slika 3.1: Primerjava dolžine niza pisav.
Vir: `http://www.lalit.org/lab/javascript-css-font-detect/`

kov. S tem je bila pogojena tudi izbira tehnologije. Glavne tehnološke komponente našega sistema so:

- vizualni del spletne strani napisan v jeziku HTML in CSS,
- program, ki na strani uporabnika izvede zajem podatkov in jih pošlje strežniškemu delu, je napisan v jeziku JavaScript,
- strežniški del sistema, ki obdela zajete podatke in jih shrani v podatkovno bazo. Napisan je v programskem jeziku PHP,
- podatkovna baza MySQL, ki skrbi za obstojnost podatkov.

Sistem sestavljata tudi dve dodatni komponenti:

- program, ki ga izvaja vtičnik Flash, je napisan v jeziku ActionScript 3, skrbi pa za zajem podatkov vtičnika Flash in seznama sistemskih pisav,
- program, ki skrbi za zajem podatkov vtičnika Silverlight, je napisan v programskem jeziku C#.

Izbira tehnologij HTML, JavaScript, Flash in C# je bila tako pravzaprav nujna za zajem želenih podatkov. PHP in MySQL pa smo si izbrali zaradi poznavanja tehnologij.

3.3 Delovanje sistema

V tem razdelku bomo podrobneje opisali delovanje našega sistema. Poglavje smo razdelili na 4 sklope: Zajem podatkov, Obdelava podatkov, Prepoznava in Shra-

njevanje podatkov.

3.3.1 Zajem podatkov

Prvi del našega sistema je program *zajem.js*, ki skrbi za zajem podatkov. Izvaja se na uporabnikovi napravi, zažene pa se ob kliku na gumb "Get my fingerprint", ki je na spletni strani. Program se izvaja v okolju JavaScript, zajame pa večino parametrov, ki smo jih opisali v poglavju 3.1.

Potek izvajanja algoritma *zajem.js* je prikazan v izseku kode 1. Kot je raz-

Izsek 1 zajem.js

```
1:  fp ← new Object()
2:  procedure STARTFP()
3:      začetni_čas ← getTime()
4:      try run async flashFonts.swf                ▷ vtičnik Flash, asinhrono
5:      try run async SilverlightData.xap           ▷ vtičnik Silverlight, asinhrono
6:      fp ← getScreenData()
7:      fp ← getNavigatorData()
8:      fp ← getTimezoneOffset()
9:      fp ← getCookies()      ▷ Preberi piškotke HTTP, Flash in Silverlight
10:     if getCookies() == null then
11:         setCookies(HTTP, Flash, Silverlight)
12:     fp ← getPluginData().
13:     fp ← detectFontsJavaScript()
14:     fp ← callback flashFonts.swf                ▷ odgovor flashFonts.swf
15:     fp ← callback SilverlightData.xap           ▷ odgovor SilverlightData.xap
16:     fp ← getTime() − začetni_čas
17:     async send obdelava.php ← fp                ▷ asinhron klic AJAX
18:     prstni_odtis ← callback obdelava.php        ▷ odgovor obdelava.php
19:     prikaži_HTML(prstni_odtis)
```

vidno iz algoritma, se vsi parametri zajema shranjujejo v objekt z imenom *fp*. Ob začetku zajema zabeležimo trenuten čas, ki služi za meritev izvajalnega časa.

Zajem se prične z zagonom vtičnikov Flash in Silverlight (vrstici 4 in 5). Tu je potrebno omeniti, da se v primeru, ko uporabnik nima omogočenih vtičnikov, vrstice 4 in 5 ter posledično tudi 14 in 15 ne izvedejo. Program se nadaljuje z zajemom podatkov o zaslonu, objektu *navigator* ter odmiku časovnega pasu. Nato zajamemo piškotke HTTP, Flash in Silverlight, ki nam bodo pomagali pri analizi rezultatov. Če ima uporabnik te piškotke nastavljene, jih pustimo take, kot so. Če pa kateri od piškotkov manjka, ga poskusimo nastaviti. V vrstici 13 pridobimo podatke o vtičnikih, v vrsticah 14 in 15 pa seznam sistemskih pisav in podatkov o vtičniku Silverlight. S tem je zajem končan, zato ponovno zabeležimo trenutni čas in izračunamo čas trajanja zajema. Nato celoten objekt *fp* pretvorimo v obliko JSON in s pomočjo klica AJAX pošljemo strežniškemu programu *obdelava.php*. Ko strežniški program z obdelavo zaključi, vrne novoustvarjeni prstni odtis programu *zajem.js*. Prstni odtis nato prikažemo uporabniku.

3.3.2 Obdelava podatkov

Kot smo že napisali, program *zajem.js* po končanem zajemu parametrov na strani uporabnika pošlje podatke v obdelavo strežniškemu delu *obdelava.php*. Strežniški del hkrati zajame tudi glave zahtevkov HTTP. Tako pridobimo vse potrebne parametre zajema in začnemo z izvajanjem algoritma *obdelava.php*

Potek izvajanja algoritma *obdelava.php* je prikazan v izseku kode 2. Na začetku algoritma zajamemo glave zahtev HTTP in sprejmemo objekt *fp*. Ustvarimo tudi tabelo *sqlData*, v katero shranjujemo obdelane podatke, ki jih bomo kasneje shranili v podatkovno bazo. V 6. vrstici s pomočjo spletne storitve `http://ipinfo.io` in naslova IP pridobimo podatek o državi uporabnika. V naslednji vrstici poiščemo podatek o arhitekturi operacijskega sistema. Če niz *User-Agent* vsebuje katerega od nizov *x86_64*, *x86-64*, *Win64*, *x64;*, *amd64*, *AMD64*, *WOW64* ali *x64_64*, označimo, da je operacijski sistem 64-biten. V nasprotnem primeru označimo, da je 32-biten. V 8. vrstici seznam pisav pridobljenih z vtičnikom Flash normaliziramo. Ugotovili smo namreč, da se seznami pisav iste naprave med brskalniki razlikujejo. Brskalniki seznamu dodajo nekaj svojih pisav, nekatere pisave pa odstranijo. Odkrili smo 38 takih pisav, zato jih pri normalizaciji odstranimo. Iz seznama odstranimo tudi duplikate ter seznam uredimo po abecednem vrstnem redu.

Izsek 2 obdelava.php

```
1: glaveZahtev ← getHTTPHeaders()
2: fp ← $_GET['fp']
3: sqlData ← new Array()
4: procedure OBDELAVA()
5:     sqlData ← glaveZahtev
6:     sqlData ← getCountry(IP)
7:     sqlData ← getCPUArchitecture(User-Agent)
8:     normFonts ← normalize_flash_fonts(flashFonts)
9:     sqlData ← pisave_fl_hash ← md5_hash(normFonts)
10:    sqlData ← pisave_js_hash ← md5_hash(jsFonts)
11:    sqlData ← zaslon_hash ← md5_hash(screenData)
12:    sqlData ← brskalnik_hash ← md5_hash(browserData)
13:    sqlData ← vticniki_hash ← md5_hash(pluginsData)
14:    sqlData ← silverlight_hash ← md5_hash(silverlightData)
15:    if device is desktop then
16:        sqlData ← naprava_hash ← md5_hash(desktopData)
17:    else
18:        sqlData ← naprava_hash ← md5_hash(mobileData)
19:    sqlData ← fp                    ▷ Shranimo tudi vse vhodne podatke
20: procedure PREPOZNAVA(naprava_hash, pisave_js_hash)
21:    if findInDatabase(naprava_hash,pisave_js_hash) then
22:        odgovor ← getAllDBRecords(naprava_hash, pisave_js_hash)
23:    odgovor ← prstni_odtis ← md5_hash(naprava_hash,pisave_js_hash)
24:    send zajem.js ← odgovor
25: procedure SHRANI(sqlData)
26:    saveToDatabase(sqlData)
```

Nato se lotimo izgradnje identifikatorjev *hash*. Le-te ustvarimo tako, da različne parametre zajema združimo in pošljemo zgoščevalni funkciji *md5*. Tako z združevanjem parametrov pridobimo enolične vrednosti *hash*, ki nam bodo kasneje služili kot identifikatorji uporabnikov. V tabeli 3.7 je z barvami prikazano, kateri parametri sestavljajo določene identifikatorje.

Vrstici 9 in 10 sta enostavni, saj seznama pisav preprosto združimo s funkcijo *implode*, ter vrednosti pošljemo funkciji *md5*. Tako pridobimo identifikatorja *pisave_fl_hash* in *pisave_js_hash*. V 11. vrstici združimo parametre zaslona (rdeča barva v tabeli 3.7). Parametra *ločljivost zaslona x* in *ločljivost zaslona y* vedno uredimo po velikosti, saj s tem preprečimo, da bi orientacija zaslona pri mobilnih napravah vplivala na vrednost identifikatorja *zaslon_hash*. 12. vrstica združi parametre brskalnika in glave zahtevka HTTP. Združimo jih in ustvarimo identifikator *brskalnik_hash*. Pomembno je poudariti, da parameter, ki vsebuje podatek o različici brskalnika, ni vključen v identifikator. To poveča odpornost identifikatorja na posodobitev brskalnika. V 13. vrstici združimo parametre vtičnikov, označene z oranžno barvo in ustvarimo identifikator *plugins_hash*. Tudi v tem identifikatorju ne uporabimo podatkov o različicah, saj se vtičniki pogosto posodabljajo. Namesto tega uporabimo le podatek o prisotnosti oziroma odsotnosti določenega vtičnika. S tem sicer izgubimo dokaj veliko količino informacije, znatno pa se poveča stabilnost identifikatorja. V 14. vrstici pa združimo parametre vtičnika Silverlight, ki so v tabeli označeni s sivo barvo.

15. vrstica je malce bolj zapletena. Zaradi različne narave mobilnih in namiznih naprav jih obravnavamo drugače. Iz niza *User-Agent* najprej razberemo tip naprave. Pri namiznih napravah je postopek enostaven, z združevanjem parametrov (rumena barva) in funkcijo *md5* ustvarimo identifikator *naprava_hash*. Pri mobilnih napravah pa združujemo malce drugačno množico parametrov (rjava barva). Kot smo že omenili, je pri mobilnih napravah niz *User-Agent* precej bolj zgovoren kot pri namiznih napravah.

Poleg tega smo ugotovili, da program *WhichBrowser.js* pogosto ne prepozna operacijskega sistema mobilne naprave. Zato smo se odločili, da med parametre vključimo podčrtani del niza *User-Agent*, ki je prikazan v spodnjem primeru 1. Poleg tega vključimo tudi parametre zaslona, *razmerje slikovnih točk* in *navigator.platform*.

Tabela 3.7

parameter	**identifikator**	
naslov IP		
država		
Referer		
X-Forwarded-For		
piškotek HTTP	temeljna	
piškotek Flash	resnica	
piškotek Silverlight	za potrebe	
piškotek ID	analize	
trajanje zajema		
čas zajema		
število pisav Flash		
število pisav JavaScript		
seznam pisav Flash	pisave_fl_hash	
seznam pisav JavaScript	pisave_js_hash	
arhitektura OS		
ime OS		
različica OS	naprava_hash	
tip naprave	(namizne naprave)	
proizvajalec naprave		
model naprave		
odmik časovnega pasu		
ločljivost zaslona x		naprava_hash
ločljivost zaslona y	zaslon_hash	(mobilne naprave)
razpoložljiva ločl. zaslona x		
razpoložljiva ločl. zaslona y		
razmerje slik. točk		
del UAS		
navigator.platform		
Accept-Language		
Accept		

– Tabela se nadaljuje na naslednji strani

Tabela 3.7 – nadaljevanje

parameter	**identifikator**	
Accept-Encoding	brskalnik_hash	
Connection		
DNT		
navigator.language		
ime brskalnika		
različica brskalnika		
jezik (Silverlight)		
jezik UI (Silverlight)		
ime OS (Silverlight)	silverlight_hash	
različica OS (Silverlight)		
število CPE (Silverlight)		
Flash		
Silverlight		
Java	vticniki_hash	
PDF		
Quicktime		
Shockwave		

Tabela 3.7: Seznam vseh zajetih parametrov. Z baravami so označeni posamezni identifikatorji

Primer 1. *Mozilla/5.0 (Linux, Android 4.3; GT-N7105 Build/JSS15J) AppleWebKit/ 537.36 (KHTML, like Gecko) Chrome/ 35.0.1916.141 Mobile Safari/ 537.36*

Omeniti velja še piškotke HTTP, Flash, Silverlight in ID, ki so označeni z modro barvo. Te piškotke nastavi program *zajem.js*, pomagali pa so nam pri analizi končnih podatkov v podatkovni bazi. Te piškotke smo uporabili kot referenčni podatek (angl. *ground truth*), na osnovi katerega smo primerjali prepoznavo s prstnim odtisom.

3.3.3 Prepoznava

Po končani obdelavi podatkov imamo sedem različnih identifikatorjev, pri čemer ima vsak svoje lastnosti in natančnosti. V prototipu našega sistema načrtno nismo združili vseh parametrov v skupni identifikator, saj želimo ugotoviti, kako natančni so posamezni identifikatorji. Drugi razlog za tak pristop pa je naša radovednost, saj želimo preveriti uspešnost sistema sledenja, ki temelji le na podlagi podatkov o napravi, brez uporabe parametrov brskalnika (angl. *cross-browser fingerprinting* oziroma *device fingerprinting*). Uporabnika želimo prepoznati tudi v primeru, ko uporablja različne brskalnike ali pa brskalnik, ki je v načinu brez beleženja zgodovine. Zaradi tega smo se odločili, da bomo v začetni implementaciji prepoznave uporabili le identifikatorja *pisave_js_hash* in *naprava_hash*. Na spletni strani `http:\fingerprinting.comyr.com` smo naš sistem predstavili kot neodvisen od brskalnika (angl. *cross-browser*), saj želimo čim več uporabnikov spodbuditi, da test zaženejo z različnimi brskalniki.

Trenutna prepoznava uporabnikov poteka takole: Z uporabo identifikatorjev *pisave_js_hash* in *naprava_hash* naredimo poizvedbo v podatkovni bazi. Če poizvedba vrne prazen seznam, uporabnik ni bil identificiran in je prepoznan kot nov uporabnik. Če poizvedba vrne seznam, kjer se identifikatorji ujemajo, je bil uporabnik prepoznan. V tem primeru zberemo podatke o njegovih predhodnih obiskih in jih pošljemo programu *zajem.js*, ki jih prikaže uporabniku. Algoritem je prikazan v izseku algoritma 2, znotraj procedure Prepoznava. S tem je postopek za uporabnika končan.

V poglavju Evalvacija in rezultati smo obravnavali tudi prepoznavo brskalnika, kjer smo analizirali vse identifikatorje ter izbrali optimalne. Končno implementacijo prepoznave lahko na podlagi rezultatov analize hitro spremenimo z zamenjavo in dodajanjem identifikatorjev.

3.3.4 Shranjevanje podatkov

V podatkovno bazo shranimo vse zajete parametre, identifikatorje ter rezultat vmesne prepoznave. Poleg tega shranimo tudi neobdelane vhodne podatke (objekt JSON), ki jih je poslal program *zajem.js*. Tako lahko algoritem kasneje tudi popravimo in prilagodimo ter ga ponovno analiziramo brez izvedbe nove raziskave.

3.4 Izvedba raziskave

Po implementaciji našega sistema smo se lotili zajema prstnih odtisov naprav realnih uporabnikov. Spletno stran `http://fingerprinting.comyr.com` smo objavili na slovenskih in tujih forumih ter socialnem omrežju Facebook. Z objavo smo povabili spletne uporabnike k sodelovanju v raziskavi, uporabniki pa so v raziskavi lahko sodelovali s klikom na gumb "Create my fingerprint". Ob kliku se je zagnal naš sistem, ki je zajel podatke o napravi, ustvaril prstni odtis naprave in ga skupaj z morebitnimi predhodnimi obiski prikazal uporabniku. Obenem je vse zajete parametre shranil v podatkovno bazo.

Poglavje 4

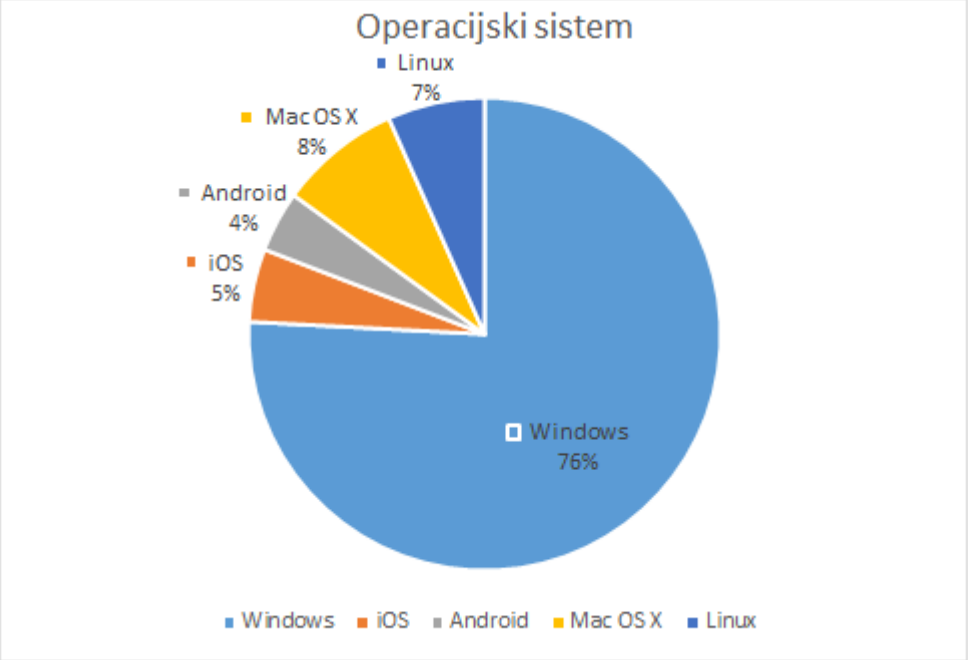
Evalvacija in rezultati

4.1 Zbrani rezultati

Zbiranje podatkov je potekalo od vključno 29. junija 2014 do 21. julija 2014, skupno torej 23 dni. V tem času smo uspeli zbrati 224 prstnih odtisov naprav. Med njimi je bilo precej uporabnikov, ki so poskus zagnali večkrat, bodisi z različnim bodisi istim brskalnikom. Zavoljo natančne analize rezultatov smo morali te poskuse prepoznati. Kot smo pisali že v poglavju Delovanje sistema 3.3, smo za ta namen uporabili piškotke HTTP, piškotke Flash ter piškotke Silverlight. S kombinacijo teh piškotkov smo lažje prepoznali isto napravo kljub temu, da je uporabnik zagnal zajem podatkov v različnih brskalnikih. Sledil je ročen pregled baze podatkov, kjer smo vsakemu zapisu v podatkovni bazi dodali atribut *device_id*, ki je služil kot temeljni identifikator naprave. Poleg omenjenih piškotkov smo si pri ročnem pregledu podatkovne baze pomagali še z naslovom IP, časom zajema ter seznamom pisav pridobljenih z metodo JavaScript. Zapisom v bazi, ki jih je naredila ista naprava vendar drug brskalnik, smo dodelili enak identifikator *device_id*.

Po pregledu podatkovne baze smo ugotovili, da je 120 različnih naprav ustvarilo 224 zajetih prstnih odtisov. Od tega je bilo 109 (91%) namiznih naprav ter 11 (9%) mobilnih. Deleži naprav so skupaj z deleži operacijskih sistemov vidni na sliki 4.1.

Pri pregledu baze smo ugotovili tudi, da je pri osmih napravah prišlo do napake pri delovanju programa *WhichBrowser.js*, saj iz niza *User-Agent* ni uspel razbrati operacijskega sistema in brskalnika naprave. To napako pripisujemo dejstvu, da



Slika 4.1: Deleži operacijskih sistemov

med raziskavo nismo posodobili podatkovne baze programa *WhichBrowser.js*, ki se posodablja tedensko. Na spletni strani `http://api.whichbrowser.net` smo namreč preverili vseh osem nizov *User-Agent* in ugotovili, da jih sistem prepozna pravilno. Ker napaka ni bila posledica naše implementacije sistema temveč zunanjega programa, smo podatke v naši podatkovni bazi ročno popravili ter ponovno zagnali algoritem za izdelavo prstnih odtisov naprav, ki je prstne odtise teh osmih naprav popravil.

4.2 Uspešnost prepoznave

Ker smo naš sistem zasnovali tako, da so nekateri identifikatorji neodvisni od brskalnika (angl. *cross-browser*) smo v nadaljevanju naredili dve ločeni analizi podatkov. V prvi analizi smo ugotavljali uspešnost prepoznave naprav (angl. *device fingerprinting*) brez identifikatorjev odvisnih od brskalnikov, v drugi analizi pa uspešnost prepoznave brskalnikov (angl. *browser fingerprinting*) z vsemi ustvarje-

nimi identifikatorji.

V obeh analizah smo ločeno obravnavali tudi namizne in mobilne naprave, saj imajo pomembne razlike. Mobilne naprave so namreč precej bolj odporne na identifikacijo s prstnim odtisom naprave, saj so posamezni modeli naprav med seboj precej podobni.

4.2.1 Prepoznava naprav

V prvi analizi smo ugotavljali uspešnost prepoznave naprav. Ker smo se omejili le na naprave, v tej analizi nismo obravnavali identifikatorjev, ki so odvisni od nastavitev brskalnika. Tako smo se omejili na identifikatorja *naprava_hash* in *pisave_js_hash*. Poleg spada tudi ločljivost zaslona, vendar pa smo tekom analize zaznali pomembno pomanjkljivost tega parametra, ki jo obravnavamo na koncu poglavja.

V tabeli 4.1 smo pokazali število različnih vrednosti identifikatorjev, ki so jih ustvarile naprave. Kot smo pričakovali, se je identifikator *pisave_js_hash* pri namiznih napravah izkazal za najbolj obetavnega, saj je 109 različnih naprav ustvarilo 96 različnih identifikatorjev. Isti identifikator je pri mobilnih napravah deloval precej slabše, saj nalaganje novih pisav na mobilne naprave ni mogoče. Opazili smo, da so imeli enaki modeli mobilnih naprav popolnoma enake sezname pisav, naprave z operacijskim sistemom iOS po 34 različnih pisav, sistemi z operacijskim sistemom Android pa 9, v dveh primerih pa še manj. Ob združitvi identifikatorjev *naprava_hash* in *pisave_js_hash* se število različnih vrednosti pri namiznih napravah malenkost poveča. Pričakovali smo malce boljši rezultat, izkazalo pa se je, da sta identifikatorja *naprava_hash* in *pisave_js_hash* medsebojno precej bolj odvisna, kot izgleda na prvi pogled. Pri mobilnih napravah združevanje identifikatorjev ni izboljšalo rezultata, saj je identifikator *pisave_js_hash* popolnoma odvisen od naprave in ne vsebuje nobene nove informacije o napravi.

V tabeli 4.2 smo prikazali števila različnih unikatnih vrednosti identifikatorjev. Unikatna vrednost identifikatorja je taka, da ustreza natanko eni napravi. To pomeni, da lahko napravo uspešno identificiramo že samo z uporabo te vrednosti identifikatorja. V tabeli je razvidno, da smo 10 namiznih naprav enolično prepoznali zgolj z uporabo identifikatorja *naprava_hash*, z uporabo *pisave_js_hash* 89 naprav, s kombinacijo obeh identifikatorjev pa 92 naprav od 109 (84%). Pri

	namizne	**mobilne**	**skupaj**
število naprav	**109**	**11**	**120**
število različnih *naprava_hash*	17	9	26
število različnih *pisave_js_hash*	96	6	102
š. r. *naprava_hash* + *pisave_js_hash*	99	9	108

Tabela 4.1: Število različnih identifikatorjev

mobilnih napravah je odstotek malce manjši; 72% vseh naprav smo prepoznali s kombinacijo obeh identifikatorjev. Pomembno je poudariti, da je število mobilnih naprav v naši množici premajhno za realno analizo in zanesljiv zaključek, zaradi narave mobilnih naprav pa predvidevamo, da bi ta odstotek z zajemom večjega števila mobilnih naprav drastično padel.

S kombinacijo obeh identifikatorjev smo torej uspeli enolično prepoznati 100 od 120 naprav (83%). V začetku smo omenili še ločljivost zaslona. Sprva smo

	namizne	**mobilne**	**skupaj**
število naprav	**109**	**11**	**120**
število unikatnih *naprava_hash*	10	8	18
število unikatnih pisave_js_hash	89	3	92
š. u. *naprava_hash* + *pisave_js_hash*	92	8	100

Tabela 4.2: Število unikatnih identifikatorjev naprav.

želeli ločljivost vključiti v prepoznavo, saj v primeru namiznih naprav prepoznavo izboljša, ima pa eno zelo slabo lastnost - nestabilnost. Ugotovili smo, da se je pri 9 napravah od 45, ki so postopek zajema opravile več kot enkrat, podatek o ločljivosti zaslona spremenil. V dveh primerih je bilo očitno, da je uporabnik želel "pretentati" sistem in je namerno spremenil ločljivost zaslona, v enem primeru pa je imel uporabnik vklopljeno povečavo (angl. *zoom*) in je zato brskalnik vračal drugačne vrednosti ločljivosti od dejanske. Dejansko ločljivost je v tem primeru še vedno možno izračunati na podlagi koeficienta povečave. V štirih primerih pa je prišlo do razlike le v parametru, ki vsebuje podatek o razpoložljivi ločljivosti zaslona, ne pa tudi v parametru ločljivost zaslona. Ta pojav pripisujemo uporabi

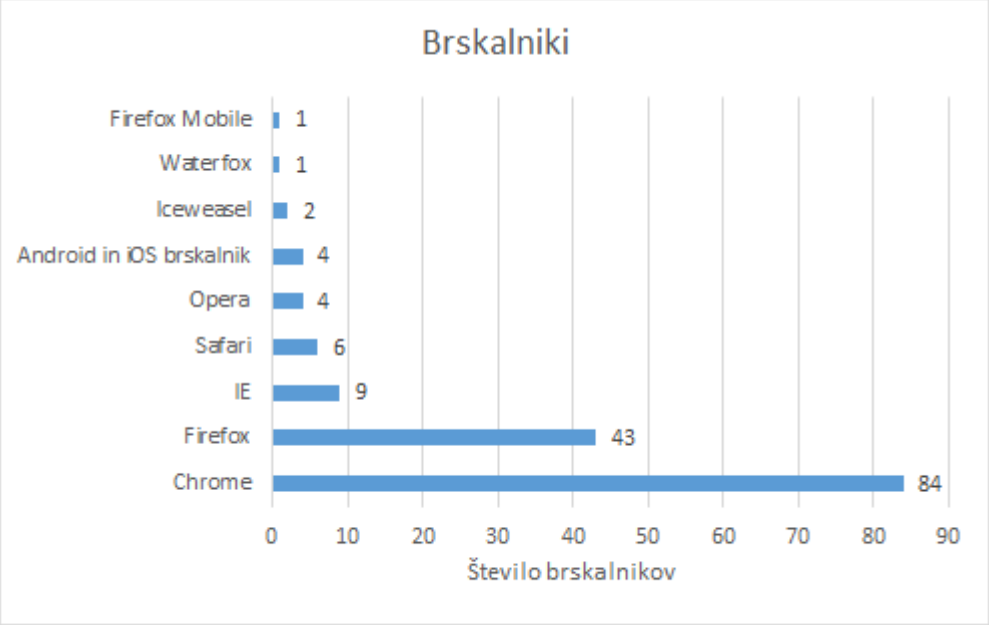
dveh enakih zaslonov hkrati, zgodi pa se, ko uporabnik odpre brskalnik v sekundarnem zaslonu, kjer ni orodne vrstice operacijskega sistema. V dveh primerih pa sta bili obe vrednosti različni, tako ločljivost zaslona kot razpoložljiva ločljivost zaslona, kar lahko pripišemo uporabi dveh različnih zaslonov. To pomeni, da je malo več kot 5 odstotkov uporabnikov namiznih računalnikov uporabljalo dva zaslona.

Ker smo v želeli preveriti še uspešnost prepoznave v kombinaciji z ločljivostjo zaslona, smo v testni podatkovni bazi popravili ločljivosti za tista dva primera, kjer je bilo očitno, da je uporabnik spremenil ločljivost, saj v realnosti do tega ne bi prišlo. Poleg tega smo popravili tudi primer, kjer je imel uporabnik nastavljeno povečavo, saj je problem trivialno rešljiv. Tako smo uspeli z uporabo identifikatorjev *naprava_hash*, *pisave_js_hash* in ločljivostjo zaslona identificirati 101 namizno napravo (93%), vendar smo zaznali tudi dva primera, ko je ista naprava ustvarila različne vrednosti identifikatorjev. Ker sta imeli dve napravi po dva različna zaslona, sta ustvarili tudi po dva različna prstna odtisa. Ta pojav krši našo ciljno lastnost sistema, ki pravi, da mora ista naprava vedno ustvariti enak prstni odtis. Vseeno je bilo 99 naprav (91%) pravilno identificiranih.

Kot je razvidno iz podatkov, je ločljivost zaslona zelo zgovorna informacija, v povezavi z razpoložljivo ločljivostjo pa še toliko bolj. Ker pa nezanemarljiv delež uporabnikov uporablja dvojni zaslon, lahko to vodi do drugačnega prstnega odtisa iste naprave. Zaradi tega dejstva podatka o ločljivosti zaslona v nadaljevanju ne uporabljamo, je pa vsekakor potencialen kandidat za nadaljnjo raziskavo.

4.2.2 Prepoznava brskalnikov

V drugi analizi smo proučili uspešnost prepoznave brskalnikov. V splošnem velja, da večina spletnih uporabnikov za vsakodnevno brskanje po spletu uporablja le en brskalnik. Ta del raziskave je zato najbolj pomemben, saj odraža realno stanje. V tej analizi smo poleg identifikatorjev naprav obravnavali tudi identifikatorje, ki so odvisni od brskalnikov. V prepoznavo smo, poleg identifikatorjev *naprava_hash* in *pisave_js_hash*, ki smo jih omenili v prejšnji analizi, dodali še identifikatorje *pisave_fl_hash*, *brskalnik_hash*, *silverlight_hash* in *vticniki_hash*. Kot smo omenili že v poglavju Obdelava podatkov 3.3.2, so v identifikatorju *brskalnik_hash* upoštevani parametri brskalnika, brez različice brskalnika. Podobno deluje identifikator *vticniki_hash*, ki vsebuje podatek o prisotnosti oziroma odsotnosti šestih



Slika 4.2: Število brskalnikov

najbolj pogostih vtičnikov, brez njihovih različic. To povečuje stabilnost omenjenih identifikatorjev in ju naredi bolj odporna na posodobitve. Identifikatorja *silverlight_hash* in *pisave_fl_hash* pa sta odvisna od vtičnikov Silverlight in Flash.

Kot smo že omenili, smo v raziskavi zbrali 224 prstnih odtisov od 120 različnih naprav. Nekaj uporabnikov teh naprav je naš program zagnalo tudi večkrat, v različnih brskalnikih. Tako smo zabeležili 154 različnih entitet brskalnikov, ki so sodelovali v raziskavi. Na sliki 4.2 smo prikazali statistične podatke brskalnikov v naši zajeti množici.

V tabeli 4.3 so prikazani statistični podatki za posamezne identifikatorje. Številke prikazujejo število različnih vrednosti identifikatorjev, ki so jih ustvarili brskalniki. Kot pričakovano, je največ, kar 108 različnih vrednosti, ustvaril identifikator *pisave_fl_hash*. To je posledica tega, da je seznam pisav, ki ga vrne vtičnik Flash, zelo specifičen in odvisen od uporabnika. Pri tem smo opazili, da se je v dveh primerih zgodilo, da je isti brskalnik ustvaril dve različni vrednosti identifikatorja *pisave_fl_hash*. V obeh primerih je uporabnik večkrat zagnal naš program, enkrat z brskalnikom v zasebnem načinu brskanja (angl. *private browsing mode*) in

	namizne	**mobilne**	**skupaj**
št. različnih brskalnikov	**141**	**13**	**154**
št. raz. *pisave_fl_hash*	107	2	108[1]
št. raz. *silverlight_hash*	20	1	20[1]
št. raz. *vtičniki_hash*	29	2	29[1]
št. raz. *brskalnik_hash*	70	11	81

Tabela 4.3: Število različnih identifikatorjev brskalnikov.

enkrat v običajnem načinu. V običajnem načinu je brskalnik vrnil celoten seznam pisav, v načinu zasebnega brskanja pa je brskalnik vrnil prazen seznam. Ta pojav smo zaznali le pri najnovejši različici vtičnika Flash, različici 14. Prav tako sta bili števili različnih vrednosti pri identifikatorjih *silverlight_hash* in *vticniki_hash* pričakovani. Pri prvem je majhno število predvsem posledica tega, da je imelo le 48 brskalnikov (31%) vključen vtičnik Silverlight. Pri drugem pa je majhno število posledica predvsem majhne količine informacije, saj identifikator *vticniki_hash* vsebuje le prisotnost oziroma odsotnost šestih najbolj pogostih vtičnikov in ne tudi njihovih različic. Presenetilo nas je visoko število različnih vrednosti identifikatorja *brskalnik_hash*, saj je 10 različnih brskalnikov (s slike 4.2) ustvarilo kar 81 različnih vrednosti. S tabele 4.3 je možno razbrati tudi, da identifikatorji *silverlight_hash*, *pisave_fl_hash* in *vticniki_hash*, ki so odvisni od vtičnikov, ne delujejo učinkovito na mobilnih napravah.

V tabeli 4.4 so prikazana števila različnih unikatnih vrednosti posameznih identifikatorjev. Ta števila torej prikazujejo število brskalnikov, ki jih lahko identificiramo le na podlagi posameznega identifikatorja. Število brskalnikov, ki jih enolično definira identifikator *pisave_fl_hash*, ni dosti manjše od števila vseh različnih vrednosti tega identifikatorja, kar je posledica raznolikosti seznama pisav med napravami. Prepoznava brskalnikov na podlagi ostalih identifikatorjev je bila malce manj uspešna. Jasno je razvidno, da identifikatorji, ki temeljijo na vtičnikih, niso uspešni pri prepoznavanju mobilnih naprav.

Do sedaj smo analizirali le prepoznavo brskalnikov s posameznimi identifi-

[1]nekatere vrednosti identifikatorjev mobilnih naprav so bile enake vrednostim identifikatorjem namiznih naprav, zato je skupno število manjše.

	namizne	**mobilne**	**skupaj**
št. različnih brskalnikov	**141**	**13**	**154**
št. unikatnih *pisave_fl_hash*	101	1	102
št. unikatnih *silverlight_hash*	11	0	11
št. unikatnih *vtičniki_hash*	13	1	13[2]
št. unikatnih *brskalnik_hash*	56	9	65

Tabela 4.4: Število unikatnih identifikatorjev brskalnikov.

katorji. S kombinacijo različnih identifikatorjev pa smo dosegli še boljšo prepoznavo brskalnikov. Za osnovo smo si izbrali identifikatorja *naprava_hash* in *pisave_js_hash*, ki smo jih predstavili že v prvi analizi.

V tabeli 4.5 smo predstavili uspešnost prepoznave s kombiniranjem različnih identifikatorjev. Z dodajanjem identifikatorjev se je v splošnem izboljšala prepoznava. Le v primeru dodajanja identifikatorja *pisave_fl_hash* je prišlo do manjšega poslabšanja prepoznave (opisano v opombi pod črto). Kot smo že omenili, sta dva uporabnika zagnala program v načinu zasebnega brskanja. Ker sta imela naložen vtičnik Flash 14, brskalnik ni vrnil seznama pisav. Zaradi tega je identifikator napačno prepoznal brskalnik. V vseh ostalih primerih je identifikator *pisave_fl_hash* deloval uspešno.

Za najboljši kombinaciji identifikatorjev sta se izkazali *NH + PJSH + VH + BH* in *NH + PJSH + VH + BH + SLH*, ki sta obe dosegli 100% uspešnost prepoznave namiznih brskalnikov in 98,7% uspešnost prepoznave mobilnih brskalnikov. Po podrobnem pregledu podatkovne baze smo ugotovili, sta imeli le dve mobilni napravi enak prstni odtis. Šlo je za dva različna mobilna uporabnika, ki sta imela popolnoma enake vrednosti vseh identifikatorjev. Oba sta uporabljala tablični računalnik iPad.

[2]ena mobilna naprava je imela enako vrednost identifikatorja kot namizna naprava.

[3]dva brskalnika sta v zasebnem načinu brskanja ustvarila dve različni vrednosti identifikatorja *pisave_fl_hash*, zato sta bila napačno prepoznana kot različna brskalnika.

	namizne		mobilne		skupaj	
NH + PJSH	92	65,2%	8	61,5%	100	64,9%
NH + PJSH + SLH	108	76,6%	8	61,5%	116	75,3%
NH + PJSH + PFLH	115[3]	81,6%	8	61,5%	123[3]	79,9%
NH + PJSH + VH	120	85,1%	8	61,5%	128	83,1%
NH + PJSH + BH	135	95,7%	11	84,6%	146	94,8%
NH + PJSH + VH + BH	141	100,0%	11	84,6%	152	98,7%
NH + PJSH + VH + BH + PFLH	139[3]	98,6%	11	84,6%	150[3]	97,4%
NH + PJSH + VH + BH + SLH	141	100,0%	11	84,6%	152	98,7%
NH+PJSH+VH+BH+PFLH+SLH	139[3]	98,6%	11	84,6%	150[3]	97,4%

Legenda: NH - *naprava_hash*, PJSH - *pisave_js_hash*, SLH - *silverlight_hash*, PFLH - *pisave_fl_hash*, VH - *vticniki_hash*, BH - *brskalnik_hash*

Tabela 4.5: Uspešnost sistema prepoznave s kombiniranjem identifikatorjev.

4.3 Evalvacija

V poglavju Implementacija (3) smo zapisali ciljne lastnosti našega sistema za prepoznavo. Na koncu smo zapisali še, da želimo preveriti uspešnost prepoznave naprave, brez uporabe parametrov, odvisnih od brskalnika (angl. *cross-browser fingerprinting*). V poglavju Prepoznava naprav (4.2.1) smo predstavili rezultate tega poskusa, kjer smo uspeli v 83% primerov pravilno prepoznati napravo. Glede na število parametrov zajema vključenih v prepoznavo je odstotek dokaj visok, vseeno pa ni dovolj visok za zanesljivo prepoznavo spletnih uporabnikov. Predvidevamo, da bi se odstotek z večjim številom zbranih prstnih odtisov še zmanjšal, precej nezanesljiv pa je tudi za sledenje mobilnim napravam. Metoda torej ni najbolj uporabna za potrebe sledenja spletnim uporabnikom, uporabna pa bi lahko bila za nekatere druge aplikacije, kot na primer preprečevanje nepooblaščenega dostopa in zlorab.

Ostale cilje smo si zastavili v kontekstu sledenja uporabnikom oziroma brskalnikom. Glavni cilj, *razvoj sistema, ki je sposoben sledenja spletnim uporabnikom na podlagi prstnega odtisa njihovega brskalnika*, smo v večji meri dosegli. V poglavju Prepoznava brskalnikov (4.2.2) smo predstavili rezultate našega sistema

prepoznave brskalnikov, kjer smo dosegli 100% uspešnost pri namiznih brskalnikih, 84,6% uspešnost pri mobilnih brskalnikih in skupno 98,7% uspešnost prepoznave vseh brskalnikov. Tu je potrebno poudariti, da je naša množica zbranih prstnih odtisov precej majhna, zato posplošitev, da bi sistem deloval enako v primeru večjega števila uporabnikov, ni mogoča. Prav tako predvidevamo, da bi se uspešnost prepoznave mobilnih naprav z večjim številom mobilnih uporabnikov znatno zmanjšala, saj smo v raziskavi ugotovili, da imajo mobilne naprave zelo majhne razlike.

Prvo ciljno lastnost, *enoličnost prstnega odtisa*, smo dosegli, saj so skoraj v vseh primerih večkratnega zajema prstnega odtisa brskalniki ustvarili enake prstne odtise. Le v dveh primerih se je pri identifikatorju *pisave_fl_hash* zgodilo, da je isti brskalnik ustvaril različen prstni odtis. Ta problem smo že opisali v poglavju Prepoznava brskalnikov (4.2.2).

Drugo ciljno lastnost, *unikatnost prstnega odtisa*, smo dosegli, saj smo ustvarili kar 98,7% unikatnih prstnih odtisov.

Tretji cilj, *robustnost sistema*, smo dosegli le deloma. Pri osmih napravah se nam je namreč zgodilo, da del sistema ni povsem pravilno deloval, problematičen pa je bil neposodobljen zunanji program *WhichBrowser.js*. S posodobitvijo smo težavo odpravili.

Četrti cilj, *hitrost sistema*, smo dosegli, saj je povprečen čas zajema trajal 3,9 sekund. Mediana vseh časov zajema je imela vrednost 2,8 sekundi.

Peti cilj, *nevidnost sistema*, smo dosegli deloma. Naš sistem zajema je uporabniku precej skrit in ni opazen neposredno. V primerih, ko uporabnik nima omogočenega katerega od vtičnikov Flash ali Silverlight, pa se pokaže majhno pojavno okno, ki uporabnika obvesti, da spletna stran potrebuje vtičnik za pravilno delovanje. Sistem je tako posredno viden le delu uporabnikov, ki vtičnikov nima omogočenih ali posodobljenih, vseeno pa bi uporabnik težko posumil, da gre za obliko sledenja. V poglavju Prepoznava brskalnikov (4.2.2) smo pokazali, da lahko uspešno prepoznamo brskalnike tudi brez uporabe vtičnikov. S kombinacijo identifikatorjev *naprava_hash*, *pisave_js_hash*, *brskalnik_hash* in *vticniki_hash*, smo dosegli 100% prepoznavo namiznih in 84,6% prepoznavo mobilnih brskalnikov. Ti identifikatorji za delovanje ne potrebujejo vtičnikov, dovolj je le okolje JavaScript. Zaradi tega tudi predlagamo omenjeno kombinacijo identifikatorjev za

končno različico našega sistema prepoznave, s tem pa bo sistem tudi popolnoma transparenten.

Zadnji, šesti cilj, *odpornost sistema na spremembe in posodobitve naprave*, smo po našem mnenju dosegli zadovoljivo. Parametre zajema, ki smo jih uporabili za naše identifikatorje, smo namreč skrbno izbrali in tako povečali odpornost sistema na spremembe. Naš sistem je ranljiv le na večje spremembe uporabnikove naprave, kot so nadgradnja operacijskega sistema, nalaganje novih pisav, nekatere spremembe v nastavitvah brskalnika ter nalaganje in onemogočanje vtičnikov. Te spremembe so pri večini uporabnikov zelo redke in zato niso tako kritične. Za podrobno analizo odpornosti našega sistema pa bi bilo potrebno izvesti dolgotrajno raziskavo, kjer bi merili spremembo prstnih odtisov skozi čas. Žal smo imeli za to premalo časa in virov.

Večino ciljev, ki smo si jih zadali, smo dosegli. Poleg naštetih ciljev, je naš sistem dosegel tudi nekaj ciljev, ki jih nismo eksplicitno navedli. Ob preizkušanju sistema smo ugotovili, da je naš sistem popolnoma odporen na izbris piškotkov, izbris zgodovine brskalnika in na način zasebnega brskanja. Poleg tega lahko naš sistem sledenja deluje preko različnih spletnih strani (angl. *cross-domain*) brez zapletene implementacije.

Poglavje 5

Zaključek

Uporabnik svetovnega spleta se dandanes težko izogne sledenju. Še težje se izogne prikritemu sledenju, saj povprečen spletni uporabnik tovrstnih tehnik ne pozna. V zadnjih letih se problemu zasebnosti na spletu posveča vse več organizacij, strokovnjakov, proizvajalcev brskalnikov in tudi zakonodajalcev. Korak v pravo smer je evropska direktiva, ki se jasno opredeljuje do sledenja uporabnikom in je med uporabniki močno okrepila ozaveščenost o ohranjanju zasebnosti na spletu. Na drugi strani pa so se predvsem zaradi potreb velikih spletnih oglaševalcev pojavile nove tehnike sledenja, ki so uporabnikom večinoma nevidne in nepoznane.

V našem diplomskem delu smo želeli pregledati področje novih tehnik sledenja in podrobno analizirati delovanje in uspešnost ene od prikritih tehnik sledenja. Prikazali in analizirali smo sistem, ki je zmožen uspešne prepoznave uporabnikov, brez uporabe piškotkov. Izdelali smo robusten sistem, ki za zajem potrebuje le okolje JavaScript in ni odvisen od vtičnikov. Slabšo prepoznavo smo dosegli le pri mobilnih napravah, ki pa je zaradi same narave mobilnih naprav pričakovana. Pomembno je poudariti, da prej omenjena evropska direktiva ureja in regulira tudi tovrstno sledenje, tako da zakonske omejitve tovrstnega sledenja že obstajajo. Problematično pa je predvsem izvajanje nadzora nad takšnim sledenjem, saj je sledenje z zajemom prstnega odtisa naprave lahko skriti. Poleg tega je problematična tudi zaščita pred takšnim sledenjem, saj enostavnih rešitev, kot jih poznamo pri običajnih piškotkih, ni. Zaradi tega je pomembno, da se namesto kurativnega preprečevanja sledenja z odkrivanjem kršiteljev ubere bolj preventivne pristope. Zakonodajalci in proizvajalci imajo namreč skupne cilje: boljšo zaseb-

nost spletnih uporabnikov. Skupaj bi lahko zastavili smernice, ki bi proizvajalcem brskalnikov predpisovale izdelavo brskalnikov, ki bodo bolj odporni proti sledenju z zajemom prstnega odtisa naprave. Kot smo že omenili v poglavju Zakonodaja in zaščita (2.4), je najboljša zaščita pred zajemom ravno ne-izstopanje iz množice. Proizvajalci brskalnikov bi lahko brez večjih sprememb znatno zmanjšali količino odtekajočih informacij iz brskalnikov ter s tem zmanjšali tudi uspešnost tovrstnega sledenja.

Omeniti pa moramo tudi dobre primere uporabe identifikacije uporabnikov z zajemom prstnega odtisa naprave. Tovrstna identifikacija je namreč lahko koristna za preprečevanje spletnih zlorab. Vse več spletnih bank, prodajalcev in drugih spletnih entitet uporablja tovrstno prepoznavo za odkrivanje nepooblaščenega dostopa do njihovih storitev. S takšno tehnologijo je mogoče izboljšati varnost spletnih storitev in posledično tudi varnost spletnih uporabnikov. Z njo je na primer mogoče zaznati poskus nepooblaščenega dostopa, preprečiti večkratno glasovanje istega uporabnika v spletnih anketah ali pa zaznati podvajanje računov spletnih storitev s strani istega uporabnika.

Tehnologija sama po sebi ni ne zlobna ne dobra. Pomembno je, kaj se z njo odločimo narediti.

Literatura

[1] A. Barth, "HTTP State Management Mechanism", RFC 6265, April 2011, dostopno na: http://tools.ietf.org/html/rfc6265

[2] D. Kristol, L. Montulli, "HTTP State Management Mechanism", RFC 2109, februar 1997, dostopno na: http://tools.ietf.org/html/rfc2109

[3] D. Kristol, L. Montulli, "HTTP State Management Mechanism", RFC 2965, oktober 2000, dostopno na: http://tools.ietf.org/html/rfc2965

[4] L. Montulli, "Persistent client state http cookies", 1994, dostopno na: http://curl.haxx.se/rfc/cookie_spec.html

[5] I. Hickson, "Web Storage", W3C, julij 2013, dostopno na: http://www.w3.org/TR/2013/REC-webstorage-20130730/

[6] A. Soltani, S. Canty, Q. Mayo, L. Thomas, C.J. Hoofnagle, "Flash Cookies and Privacy", avgust 2009, dostopno na: http://ssrn.com/abstract=1446862

[7] M. Ayenson, D.J. Wambach, A. Soltani, N. Good, C.J. Hoofnagle, "Flash Cookies and Privacy II: Now with HTML5 and ETag Respawning", julij 2011, dostopno na: http://ssrn.com/abstract=1898390

[8] S. Kamkar, "evercookie – never forget", september 2010, dostopno na http://samy.pl/evercookie/

[9] "About open tracking", Mailchimp, maj 2014, dostopno na: http://kb.mailchimp.com/article/about-open-tracking/, (1.6.2014)

[10] P. Eckersley, "How online tracking companies know most of what you do online (and what social networks are doing to help them)", Electronic frontier foundation, september 2009, dostopno na: https://www.eff.org/deeplinks/2009/09/online-trackers-and-social-networks, (1.6.2014)

[11] K. Boda, Á.M. Földes, G.Gy. Gulyás, S. Imre, "User Tracking on the Web via Cross-Browser Fingerprinting", 16th Nordic Conference in Secure IT Systems (Nordsec 2011), oktober 2011, Talinn, Estonia, dostopno na: http://pet-portal.eu/files/articles/2011/fingerprinting/cross-browser_fingerprinting.pdf, (9.7.2014)

[12] P. Eckersley, "How Unique Is Your Web Browser?", Privacy Enhancing Technologies, 2010, dostopno na: http://panopticlick.eff.org/browser-uniqueness.pdf, (10.5.2014)

[13] L. Sweeney, "Computational disclosure control, A Primer on Data Privacy Protection", 2001, dostopno na: http://groups.csail.mit.edu/mac/classes/6.805/articles/privacy/sweeney-thesis-draft.pdf, (13.7.2014)

[14] P. Eckersley, "A Primer on Information Theory and Privacy", 2010, dostopno na: https://www.eff.org/deeplinks/2010/01/primer-information-theory-and-privacy, (13.7.2014)

[15] H. Tillman, "Browser Fingerprinting: Tracking ohne Spuren zu hinterlassen", diplomsko delo, 2013, dostopno na: http://bfp.henning-tillmann.de/downloads/Henning%20Tillmann%20-%20Browser%20Fingerprinting.pdf, (14.7.2014)

[16] R. Broenink, "Using Browser Properties for Fingerprinting Purposes", 16th Twente Student Conference on IT, 27. januar 2012, dostopno na: http://www.letmetrackyou.org/paper.pdf, (14.7.2014)

[17] K. Mowery, et al., "Fingerprinting information in JavaScript implementations", Proceedings of W2SP (2011), 2011, dostopno na: http://cseweb.ucsd.edu/~hovav/dist/jspriv.pdf, (14.7.2014)

[18] K. Mowery, S. Hovav, "Pixel perfect: Fingerprinting canvas in HTML5", Proceedings of W2SP (2012), 2012, dostopno na: `http://cseweb.ucsd.edu/~hovav/dist/canvas.pdf`, (14.7.2014)

[19] T. Kohno, A. Broido, K.C. Claffy, "Remote physical device fingerprinting", Dependable and Secure Computing, IEEE Transactions on 2.2, 93-108, 2005, dostopno na: `http://www.caida.org/publications/papers/2005/fingerprinting/KohnoBroidoClaffy05-devicefingerprinting.pdf`, (14.7.2014)

[20] G. Acar, et al., "FPDetective: Dusting the web for fingerprinters", Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security ACM, 2013, dostopno na: `http://www.cosic.esat.kuleuven.be/publications/article-2334.pdf`, (14.7.2014)

[21] B. Prastalo, C. Sganga, "Internet privacy in the context of online advertising", 2014, dostopno na: `http://www.etd.ceu.hu/2014/prastalo_boris.pdf`, (16.7.2014)

[22] D.L. Baumer, J.B. Earp, J.C. Poindexter, "Internet privacy law: A comparison between the United States and the European Union.", Computers & Security 23.5, strani 400-412, 2004, dostopno na: `http://www4.ncsu.edu/~baumerdl/C%26S_legalComparisonEUvsUS_forJC.doc`, (16.7.2014)

[23] D.J. Solove, M. Rotenberg, P.M. Schwartz, "Information privacy law", 2003, dostopno na: `http://docs.law.gwu.edu/facweb/dsolove/information-privacy-law/files/ipl-update-2007.pdf`, (16.7.2014)

[24] R. Singel, "Bill to Restrict Online Tracking Introduced in Congress", Wired.com, 2011, dostopno na: `http://www.wired.com/2011/02/do-not-track-bill/`, (16.7.2014)

[25] Evropski parlament, "Directive 95/46/EC", Official Journal L 281, strani 31-50, 23.11.1995, dostopno na: `http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31995L0046:en:HTML`, (16.7.2014)

[26] Evropski parlament, "Directive 2002/58/EC", Official Journal L 201, strani 37-47, 31.07.2002, dostopno na: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:32002L0058:en:HTML, (16.7.2014)

[27] Evropski parlament, "Directive 2009/136/EC", Official Journal L 337/11, 18.12.2009, dostopno na: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2009:337:0011:0036:en:PDF, (16.7.2014)

[28] Državni zbor Republike Slovenije, "Zakon o elektronskih komunikacijah (ZEKom-1)", Uradni list RS, št. 109/2012, 31.12.2012, dostopno na: http://www.uradni-list.si/1/content?id=111442, (16.7.2014)

[29] Informacijski pooblaščenec RS, "Smernice Informacijskega pooblaščenca", 2013, dostopno na: https://www.ip-rs.si/fileadmin/user_upload/Pdf/smernice/Smernice_o_uporabi_piskotkov.pdf, (16.7.2014)

[30] Ed-PR, "European Data Protection Round Table", dostopno na: http://ec.europa.eu/justice/news/consulting_public/0003/contributions/organisations_not_registered/edpr_en.pdf, (16.7.2014)

[31] Data protection working party, "Opinion 16/2011 on EASA/IAB Best Practice Recommendation on Online Behavioural Advertising", Article 29, 2011, dostopno na: http://ec.europa.eu/justice/data-protection/article-29/documentation/opinion-recommendation/files/2011/wp188_en.pdf, (16.7.2014)

[32] ICO, "Guidance on the rules on use of cookies and similar technologies", 2012, dostopno na: https://ico.org.uk/~/media/documents/library/Privacy_and_electronic/Practical_application/cookies_guidance_v3.pdf, (16.7.2014)